lee215 ★ 47716 Last Edit: August 24, 2019 11:03 PM 4.1K VIEWS

112

Intuition

I take it this way:
We cannot build any well.
There is one and only one hidding well in my house (house 0).
The cost to lay pipe between `house[i]` and my house is `wells[i]`.

In order to supply water to the whole village,
we need to make the village a coonected graph.

Explanation

Merge all costs of pipes together and sort by key.
Greedily lay the pipes if it can connect two seperate union.
Appply union find to record which houses are connected.

Complexity

Time `O(ElogE)`
Space `O(N)`

C++

```
vector<int> uf;
int minCostToSupplyWater(int n, vector<int>& wells, vector<vector<int>>& pipes) {
    uf.resize(n + 1, 0);
    for (auto& p : pipes) swap(p[0], p[2]);
    for (int i = 0; i < n; ++i) {
        uf[i + 1] = i + 1;
        pipes.push_back({wells[i], 0, i + 1});
    }
    sort(pipes.begin(), pipes.end());

    int res = 0;
    for (int i = 0; n > 0; ++i) {
        int x = find(pipes[i][1]), y = find(pipes[i][2]);
        if (x != y) {
            res += pipes[i][0];
            uf[x] = y;
            --n;
        }
    }
    return res;
}

int find(int x) {
    if (x != uf[x]) uf[x] = find(uf[x]);
    return uf[x];
}
```

Python:

```
def minCostToSupplyWater(self, n, wells, pipes):
    uf = {i: i for i in xrange(n + 1)}

    def find(x):
        if x != uf[x]:
            uf[x] = find(uf[x])
        return uf[x]

    w = [[c, 0, i] for i, c in enumerate(wells, 1)]
    p = [[c, i, j] for i, j, c in pipes]
    res = 0
    for c, x, y in sorted(w + p):
        x, y = find(x), find(y)
        if x != y:
            uf[find(x)] = find(y)
            res += c
            n -= 1
        if n == 0:
            return res
```

Java

```
    List<int[]> edges = new ArrayList<>();
    for (int i = 0; i < n; i++) {
        uf[i + 1] = i + 1;
        edges.add(new int[] {0, i + 1, wells[i]});
    }
    for (int[] p : pipes) {
        edges.add(p);
    }
    Collections.sort(edges, (a, b) -> Integer.compare(a[2], b[2]));

    int res = 0;
    for (int[] e : edges) {
        int x = find(e[0]), y = find(e[1]);
        if (x != y) {
```

```
        }
    return res;
}

private int find(int x) {
    if (x != uf[x]) uf[x] = find(uf[x]);
    return uf[x];
}
```

Comments: 28 Best Most Votes Newest to Oldest Oldest to Newest

Type comment here... (Markdown is supported)

Post

wl2929 ★ 139 September 4, 2019 12:40 PM

我要是能自己想出这方法我宁愿上街裸奔

64 Reply

yuanb10 ★ 664 Last Edit: August 24, 2019 10:04 PM

Java version. Gosh, this is so lengthy compared to Python.

```
class Solution {
    class UnionFind {
        int[] parent;

        public UnionFind(int n) {
            parent = new int[n];
            for (int i = 0; i < n; i++) {
                parent[i] = i;
```

Read More

10 Reply

user0886X ★ 5 October 23, 2019 9:23 PM

In Java solution, why do we need `--n`?

4 Show 1 reply Reply

stockfish ★ 169 August 24, 2019 9:39 PM

sort the edges is O(|E|log(|E|))

4 Show 1 reply Reply

xiaozhi1 ★ 43 October 14, 2019 11:54 PM

How can you come up with the idea of "changing cost of wells to pipes"! Such a genius (or alien?)

3 Reply

copyfriend ★ 27 August 25, 2019 11:35 AM

I think we can do uf[x] = y instead of uf[find(x)] = find(y). But skr, 666, old iron unbeatable!

3 Show 1 reply Reply

stockfish ★ 169 August 24, 2019 9:49 PM

感觉现在MST是竞赛标配了。。。

4 Show 1 reply Reply

frankliguanju ★ 22 December 13, 2019 1:30 AM

Why are you so smart and I am so stupid???

2 Reply

jmzhang18 ★ 22 February 4, 2020 12:07 PM

OMG, using the '0' is genius

1 Reply

jaithrik ★ 28 January 15, 2020 2:40 PM

I cannot see why Minimal spanning tree would be the solution here? I was trying DP (or recursive memoize like take a well or not). Can someone guide here? @lee215 maybe but I'm sure you get many requests like this one.

1 Reply