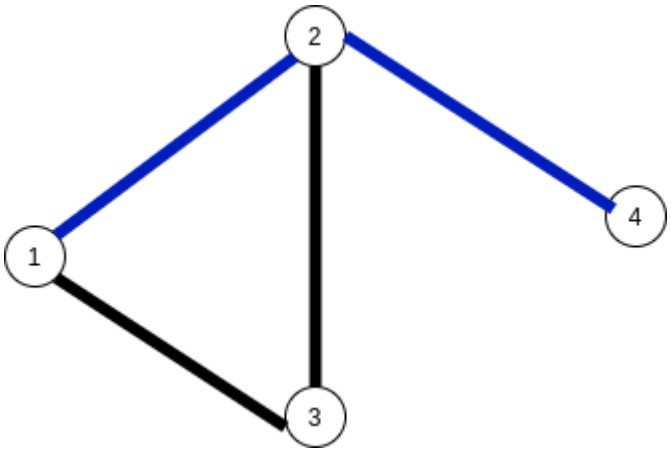

Pak Dengklek Mudik

Deksripsi Singkat

Diberikan kota yang dipresentasikan sebagai vertex dan jalan yang direpresentasikan sebagai edge. Jalan terdiri atas dua jenis yakni jalan khusus yang membutuhkan satu tiket untuk melaluinya dan jenis jalan umum yang tidak membutuhkan tiket untuk melaluinya. Diketahui terdapat Q orang dimana setiap orang mempunyai jumlah tiket tertentu, kota awal, dan kota tujuan masing-masing. Tentukan untuk masing-masing orang apakah bisa mudik atau tidak.

Contoh Ide untuk Soal Non Bonus

Problem ini pada dasarnya merupakan *graph traversal* dengan sedikit modifikasi. Oleh karena itu, untuk menyelesaikan problem ini, kita bisa menggunakan DFS atau BFS dengan sedikit modifikasi. Pada *graph traversal* standar, state untuk menyatakan apakah node sudah pernah dikunjungi hanya berupa indeks dari node itu sendiri. Untuk problem ini hal tersebut tidak bisa dilakukan karena outcome dari kunjungan yang mempunyai tiket sebanyak a bisa saja berbeda dengan kunjungan yang mempunyai tiket sebanyak b dengan tujuan node yang sama. Berikut merupakan contoh ilustrasi untuk menggambarkan hal tersebut.



Katakanlah kita mengecek apakah bisa mengunjungi node 4 dari node 1 apabila diberikan tiket sebanyak 1. Katakanlah dilakukan DFS. Pada kemungkinan awal, apabila kita hanya menggunakan state node saja untuk visited, jalur awal visitasi adalah 1-2 (4 tidak masuk karena tidak ada tiket yang bisa digunakan lagi), maka node 1 dan node 2 akan ditandai sebagai visited. Lalu dilanjutkan visitasi node dari node 3. Apabila kita hanya menggunakan node saja untuk state visited dari suatu node, maka node 2 tidak bisa dikunjungi dari 3 dikarenakan node 2 sudah pernah ditandai dari visitasi awal sehingga berakibat node 4 tidak bisa dikunjungi dari node 1 yangmana semestinya dengan 1 tiket, node 4 bisa dikunjungi dari node 1. Untuk mengatasi hal tersebut, kita bisa menambahkan parameter berupajumlah tiket yang tersisa selain dari indeks node untuk state visited dari masing-masing node. Ide solusi ini mempunyai *worst case time complexity* per query-nya ***O((V+E)K))*** dan *overall worst case time complexity* **O((V+E)KQ))**.

Berikut ini adalah potongan program untuk implementasi ide solusi di atas.

```
//solusi bfs
public int bfs(int initialSrc, int dst, int ticket, boolean [][] visited)
{
        Queue<Pair> frontier = new LinkedList<>();
        frontier.add(new Pair(initialSrc, ticket));
        while(!frontier.isEmpty()) {
            Pair current = frontier.remove();
            int src = current.first;
            int remTicket = current.second;
            if (src==dst) return 1;
            visited[src][remTicket] = true;
            for (Edge edge: edgeList[src]) {
                int nextHop = edge.to;
                int edgeType = edge.type;
                if (!visited[nextHop][remTicket]) {
                    if (edgeType==1 && remTicket > 0)
                        frontier.add(new Pair(nextHop, remTicket-1));
                    else if(edgeType==0)
                        frontier.add(new Pair(nextHop, remTicket));
                }
            }
        }
        return 0;
    }

//solusi dfs
public int dfs(int src, int dst, int ticket, boolean[][] visited) {
        if (src==dst)
            return 1;
        if(visited[src][ticket]) return 0;
        visited[src][ticket] = true;
        for (Edge edge: edgeList[src]) {
            int nextHop = edge.to;
            int edgeType = edge.type;
            int ret=0;
            if (!visited[nextHop][ticket]) {

                if (edgeType==1 && ticket > 0)
                    ret = dfs(nextHop, dst, ticket-1, visited);
                else if (edgeType==0)
                   ret = dfs(nextHop, dst, ticket, visited);
            }
            if (ret==1) return 1;
        }
        return 0;
    }
```

Contoh Solusi Bonus:

Ada dua ide yang mungkin:

Ide 1: kompresi graf

Observasi disini bahwa dua buah kota yang dapat dicapai dengan hanya menggunakan jalan biasa dapat dianggap satu kesatuan karena dapat berpindah dari dua kota tersebut tanpa batasan tertentu. Jadi algoritmanya:

1. Partisi graf menjadi beberapa subgraf yang dimana semua kota pada suatu subgraf dapat dicapai satu sama lain dengan jalan biasa
2. Buat graf baru dimana hanya ada 1 node perwakilan untuk setiap subgraf. Sebua edge di subgraf ini memiliki bobot 1
3. Hitung all pair shortest path di graf baru ini dengan BFS biasa. Katakan F(U, V) sebagai jarak U dan V di graf baru ini.
4. Dua buah vertex A dan B bisa dicapai dengan tiket T jika F(A, B) <= T

Ide 2: BFS 0/1

Permasalahan ini bisa dipandang sebagai permasalahan shortest path, dengan jalan tol berbobot 1, dan jalan biasa berbobot 0. Kota A bisa ke B jika dan hanya jika shortest path A ke B <= T.

Kita bisa menghitung ini dengan dijkstra, namun akan mendapatkan TLE karena O(E log V). Untuk mengoptimasi, kita bisa menggunakan algoritma BFS 0/1, sebuah modifikasi sederhana BFS yang dimana kompleksitasnya O(V + E): https://cp-algorithms.com/graph/01_bfs.html

Secara sederhana, pada BFS 0/1 ada dua kemungkinan queue:

- Jika next edgenya 0, maka push ke depan queue
- Jika next edgenya 1, maka push ke belakang queue