II OIFem (2022)

Final: Día 1

Soluciones

Olimpiada Informática Femenina

Comunicación encriptada

Autora del problema: Blanca Huergo Muñoz

Teoría

- Strings
- Bucles

Solución

Imprimimos primero los carácteres en índices que den resto 0 al dividir entre 10, luego resto 1, luego resto 2 y así hasta llegar a resto 9.

Código- C++

```cpp
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;


int main() {
    int t;
    cin >> t;
    while (t--) {
        string s;
        cin >> s;
        for (int i = 0; i < 10; ++i) {
            for (int j = i; j < (int) s.size(); j += 10) {
                cout << s[j];
            }
        }
        cout << endl;
    }
}
```

Código- Python

```python
t = int(input())
for x in range(t):
    S = input()
    filas = ["" for i in range(10)]
    for i in range(len(S)):
        filas[i%10] += S[i]
    print("".join(filas))
```

Hackovid

Autor del problema: Martín Sánchez-Pedreño Sánchez

Teoría

- Multi-source BFS

Solución

Iniciamos una BFS desde todos los positivos a la vez, marcando los contactos directos y adyacentes de esta forma. Hay que cuidar mucho la implementación para asegurarse de que si una persona entra en más de una categoría, escojamos la que demuestre un grado de perjuicio mayor.

Código- C++

```cpp
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
#include <queue>
using namespace std;

int n, p;
vector<int> positivos;
vector<vector<int>> adjList;

void solve() {
    queue<int> Q;
    vector<int> estado(n, 0);
    for (int i = 0; i < p; i++) {
        estado[positivos[i]] = 3;
        Q.push(positivos[i]);
    }
    while(Q.size()) {
        int cur = Q.front();
        Q.pop();
        if (estado[cur] == 3) {
            for (int & contacto: adjList[cur]) {
                if (estado[contacto] == 0) {
                    estado[contacto] = estado[cur]-1;
                    Q.push(contacto);
                }
            }
        } else {
            // estado[cur] = 2
            for (int & contacto: adjList[cur]) {
                if (estado[contacto] == 0) {
                    estado[contacto] = estado[cur]-1;
                }
            }
        }
    }
```

```
    cout << estado[0];
    for (int i = 1; i < n; i++)
        cout << ' ' << estado[i];
    cout << '\n';
}

int main() {
    int t, m;
    cin >> t;
    while(t--) {
        cin >> n >> m >> p;
        positivos = vector<int>(p);
        for (int i = 0; i < p; i++)
            cin >> positivos[i];
        adjList = vector<vector<int>>(n);
        for (int i = 0; i < m; i++) {
            int a, b;
            cin >> a >> b;
            adjList[a].push_back(b);
            adjList[b].push_back(a);
        }
        solve();
    }
    return 0;
}
```

Código- Python

```
INF = 10**8
t = int(input())
for _ in range(t):
    n, m, p = map(int, input().split())
    posi = []
    if p > 0:
        posi = list(map(int, input().split()))
    adj = [[] for _ in range(n)]
    for _ in range(m):
        u, v = map(int, input().split())
        adj[u].append(v)
        adj[v].append(u)
    dist = [INF for _ in range(n)]
    q = []
    for u in posi:
        q.append(u)
        dist[u] = 0
    while len(q) > 0:
        u = q.pop(0)
        for v in adj[u]:
            if dist[v] == INF:
                dist[v] = dist[u] + 1
                q.append(v)
    ans = [3 - min(3, dist[i]) for i in range(n)]
    print(' '.join(map(str, ans)))
```

Bloq mayus

Autor del problema: Ferran Alet Puig

Teoría

- Programación dinámica

Solución

Podemos resolver este problema utilizando dos DPs: `mayuscula`, con el mínimo coste de llegar al i-ésimo carácter (indexando por 1) acabando en modo mayúscula en `mayuscula[i]` y `minuscula`, con el mínimo coste de llegar al i-ésimo carácter (indexando por 1) acabando en modo minúscula en `minuscula[i]`.

Código- C++

```cpp
#include <iostream>
#include <vector>
#include <string>

using namespace std;

int main() {
  int casos;
  cin >> casos;
  for(int caso = 0; caso < casos; caso++) {
    int C,A,O;
    cin >> C>>A>>O;
    string s;
    cin >> s;
    int n = s.size();
    int INF = 1e9;
    vector<int> minuscula(n+1, INF);
    vector<int> mayuscula(n+1, INF);
    minuscula[0] = 0;
    for(int i=0;i<n;i++){
      if(s[i]>='a' and s[i]<='z') { // minuscula
        minuscula[i+1] = min(minuscula[i+1], minuscula[i] + A); // Estaba en
        ↪ minuscula y tardo A
        minuscula[i+1] = min(minuscula[i+1], mayuscula[i] + C + A); // Estaba en
        ↪ mayuscula, cambio a minuscula y tardo C+A
        mayuscula[i+1] = min(mayuscula[i+1], mayuscula[i] + O); // Estaba en
        ↪ mayuscula, me mantengo y tardo O
      }
      else { // mayuscula
        mayuscula[i+1] = min(mayuscula[i+1], mayuscula[i] + A); // Estaba en
        ↪ mayuscula y tardo A
        mayuscula[i+1] = min(mayuscula[i+1], minuscula[i] + C + A); // Estaba en
        ↪ minuscula, cambio a mayuscula y tardo C+A
        minuscula[i+1] = min(minuscula[i+1], minuscula[i] + O); // Estaba en
        ↪ minuscula, me mantengo y tardo O
      }
    }
```

```cpp
    // Calcular coste si no puedes cambiar a modo mayuscula. 'A' por minusculas,
    ↪  'O' por mayusculas.
    int solominuscula = 0;
    for(int i=0;i<n;i++){
      if(s[i]>='a' and s[i]<='z') {
        solominuscula += A;
      }
      else {
        solominuscula += O;
      }
    }
    cout << solominuscula - min(minuscula[n], mayuscula[n]) << endl;
  }
}
```

Código- Python

```python
import sys
t = int(sys.stdin.readline())
for xx in range(t):
    C, A, O = map(int, sys.stdin.readline().split())
    s = sys.stdin.readline().strip('\n')
    n = len(s)
    INF = 10**9
    minuscula = [INF for i in range(n+1)]
    mayuscula = [INF for i in range(n+1)]
    minuscula[0] = 0
    for i in range(n):
        if s[i] >= 'a' and s[i] <= 'z':
            minuscula[i+1] = min(minuscula[i+1], minuscula[i] + A)
            minuscula[i+1] = min(minuscula[i+1], mayuscula[i] + C + A)
            mayuscula[i+1] = min(mayuscula[i+1], mayuscula[i] + O)
        else:
            mayuscula[i+1] = min(mayuscula[i+1], mayuscula[i] + A)
            mayuscula[i+1] = min(mayuscula[i+1], minuscula[i] + C + A)
            minuscula[i+1] = min(minuscula[i+1], minuscula[i] + O)
    solominuscula = 0
    for i in range(n):
        if s[i] >= 'a' and s[i] <= 'z':
            solominuscula += A
        else:
            solominuscula += O
    print(solominuscula - min(minuscula[n], mayuscula[n]))
```

Puentes en ruinas

Autor del problema: Max Balsells I Pàmies

Teoría

- UFDS

Solución

Podemos resolver este problema haciendo UFDS en orden inverso de caídas, es decir, inicialmente construimos los conjuntos correspondientes a los puentes que no llegan a caer y vamos añadiendo puentes en orden inverso de caídas, en todo momento apuntando la respuesta con ese número de puentes caídos. Finalmente, imprimimos las respuestas en orden inverso.

Código- C++

```cpp
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
#include <set>
using namespace std;
int n, m, q;
vector<int> r, p;

int get_parent(int a) {
    if(p[a] == a) {
        return a;
    }
    return p[a] = get_parent(p[a]);
}

void check() {
    set<int> parents;
    for(int i = 0; i < n; i++) {
        parents.insert(get_parent(i));
    }
    int total = 0;
    for(int i : parents) {
        total += r[i];
    }
}

void merge(int a, int b) {
    a = get_parent(a);
    b = get_parent(b);
    if(a == b) {
        return;
    }
    if(r[a] > r[b]) {
        p[b] = a;
        r[a] += r[b];
```

```
    } else {
        p[a] = b;
        r[b] += r[a];
    }
}

int main() {
    int t;
    cin >> t;

    while(t--) {
        cin >> n >> m;

        vector<pair<int,int> > puentes(m);
        vector<bool> puente_borrado(m, false);

        for(int i = 0; i < m; i++) {
            int x, y;
            cin >> x >> y;
            puentes[i] = make_pair(x, y);
        }

        cin >> q;

        vector<int> idxs(q);

        for(int i = 0; i < q; i++) {
            int idx;
            cin >> idx;
            idxs[i] = idx;
            puente_borrado[idx] = true;
        }

        p.resize(n);
        r.resize(n);

        for(int i = 0; i < n; i++) {
            p[i] = i;
            r[i] = 1;
        }

        int ans = 1;

        for(int i = 0; i < m; i++) if(!puente_borrado[i]) {
            int x = puentes[i].first, y = puentes[i].second;
            merge(x, y);
            ans = max(ans, r[get_parent(x)]);
        }

        check();

        vector<int> out_ans;

        out_ans.push_back(ans);

```

```
        for(int i = q - 1; i >= 0; i--) {
            int idx = idxs[i];
            int x = puentes[idx].first, y = puentes[idx].second;
            merge(x, y);
            ans = max(ans, r[get_parent(x)]);
            out_ans.push_back(ans);
        }

        for(int i = q; i >= 0; i--) {
            if(i != q) {
                cout << " ";
            }
            cout << out_ans[i];
        }

        cout << endl;
    }

    return 0;
}
```

Código- Python

```
def get_parent(a, p):
    if p[a] == a:
        return a
    p[a] = get_parent(p[a], p)
    return p[a]

def merge(a, b, p, r):
    a = get_parent(a, p)
    b = get_parent(b, p)
    if a == b:
        return
    if r[a] > r[b]:
        p[b] = a
        r[a] += r[b]
    else:
        p[a] = b
        r[b] += r[a]

t = int(input())

for tt in range(t):
    line = input().split(" ")
    n = int(line[0])
    m = int(line[1])

    puentes = []
    puente_borrado = [False for i in range(m)]

    for i in range(m):
        line = input().split(" ")
        x = int(line[0])
```

```
        y = int(line[1])
        puentes.append((x, y))

    line = input().split(" ")
    q = int(line[0])

    idxs = []

    for i in range(q):
        idx = int(line[i + 1])
        puente_borrado[idx] = True
        idxs.append(idx)

    p = [i for i in range(n)]
    r = [1 for i in range(n)]

    ans = 1

    for i in range(m):
        if puente_borrado[i] == True:
            continue
        merge(puentes[i][0], puentes[i][1], p, r)
        ans = max(ans, r[get_parent(puentes[i][0], p)])

    out_ans = [ans]

    for i in range(q)[::-1]:
        idx = idxs[i]
        merge(puentes[idx][0], puentes[idx][1], p, r)
        ans = max(ans, r[get_parent(puentes[idx][0], p)])
        out_ans.append(ans)

    for i in range(q + 1)[::-1]:
        print(out_ans[i], end=" ")
    print("")
```

Lista de máximos

Autora del problema: Blanca Huergo Muñoz

Teoría

- Stacks
- Programación dinámica

Solución

Este problema se podía resolver en $O(n \log n)$, aunque la mejor solución era una lineal. Primero usamos stacks para, para todo i entre 0 y $n-1$, encontrar `indIzq[i]` y `indDer[i]`, el primer índice a la izquierda y la derecha respectivamente tal que `a[ind] > a[i]`. Por lo tanto, sabemos que `a[i]` es el máximo de `a[indIzq[i]+1 ... i ... a[indDer[i]]` y que este es el mayor bloque del que el índice i es el máximo. Además, esto nos dice que para todas las longitudes entre 1 y `indDer[i] - indIzq[i] - 1` hay un bloque del que `a[i]` es el máximo. Partiendo de esto, podemos calcular la respuesta.

Código- C++

```cpp
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
#include <stack>
using namespace std;

void solve(vector<int> & nums) {
    stack<int> stackIzq;
    vector<int> indIzq(nums.size(), -1);
    stackIzq.push(0);
    for (int i = 1; i < (int) nums.size(); i++) {
        while(stackIzq.size() && nums[stackIzq.top()] <= nums[i])
            stackIzq.pop();
        if (stackIzq.size())
            indIzq[i] = stackIzq.top();
        stackIzq.push(i);
    }

    stack<int> stackDer;
    vector<int> indDer(nums.size(), nums.size());
    stackDer.push(nums.size()-1);
    for (int i = (int) nums.size()-2; i >= 0; i--) {
        while(stackDer.size() && nums[stackDer.top()] <= nums[i])
            stackDer.pop();
        if (stackDer.size())
            indDer[i] = stackDer.top();
        stackDer.push(i);
    }

    vector<int> ans(nums.size(), 1e9);
    for (size_t i = 0; i < nums.size(); i++) {
```

```
        int len = indDer[i] - indIzq[i] - 1;
        ans[len-1] = min(ans[len-1], nums[i]);
    }

    int maximoActual = 1e9;
    for (int i = (int) nums.size() - 1; i >= 0; i--) {
        ans[i] = min(ans[i], maximoActual);
        maximoActual = ans[i];
    }

    cout << ans[0];
    for (size_t i = 1; i < ans.size(); i++)
        cout << ' ' << ans[i];

    cout << endl;
}


int main() {
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    int t, n;
    cin >> t;
    while(t--) {
        cin >> n;
        vector<int> nums(n);
        for (int i = 0; i < n; i++)
            cin >> nums[i];
        solve(nums);
    }
    return 0;
}
```

Código- Python

```
t = int(input())
for xx in range(t):
    nums = list(map(int, input().split()))
    nums = nums[1:]
    stackIzq = [0]
    indIzq = [-1 for i in range(len(nums))]
    for i in range(1, len(nums)):
        while(len(stackIzq) and nums[stackIzq[-1]] <= nums[i]):
            stackIzq.pop()
        if len(stackIzq):
            indIzq[i] = stackIzq[-1]
        stackIzq.append(i)

    stackDer = [len(nums)-1]
    indDer = [len(nums) for i in range(len(nums))]
    for i in range(len(nums)-2, -1, -1):
        while len(stackDer) and nums[stackDer[-1]] <= nums[i]:
            stackDer.pop()
        if len(stackDer):
```

```
            indDer[i] = stackDer[-1]
        stackDer.append(i)

    ans = [10**9 for i in range(len(nums))]
    for i in range(len(nums)):
        length = indDer[i] - indIzq[i] - 1
        ans[length-1] = min(ans[length-1], nums[i])

    maximoActual = 10**9
    for i in range(len(nums)-1, -1, -1):
        ans[i] = min(ans[i], maximoActual)
        maximoActual = ans[i]

    ans = list(map(str, ans))
    print(" ".join(ans))
```

Caperucita roja

Autor del problema: Félix Moreno Peñarrubia

Teoría

- Dijkstra
- Programación dinámica

Solución

Generamos el árbol de Dijkstra desde la casa de la abuela. A partir de él, hacemos una DP con NK estados, donde cada valor de la dp es el máximo tiempo cogiendo K aristas que no pertenecen al árbol.

Código- C++

```cpp
#include <iostream>
#include <vector>
#include <map>
#include <algorithm>
#include <set>
#include <queue>
#include <complex>
#include <sstream>
#include <assert.h>

using namespace std;
typedef unsigned long long int ull;
typedef long long int ll;
typedef vector <int> vi;
typedef vector <ll> vll;
typedef vector <vi> vvi;
typedef vector <vvi> vvvi;
typedef vector <vvvi> vvvvi;
typedef pair <int, int> ii;
typedef vector <ii> vii;
typedef vector <vii> vvii;
typedef pair <int, ii> tri;

vvii G;
vvi dp;
vi T;
vi C;
int n, m, k;

void dijkstra(){
    priority_queue <vi> Q;
    Q.push({-0, n-1, -1});
    vi visited(n, 0);

    while (not Q.empty()){
        vi aux = Q.top(); Q.pop();

```

```
        int d = -aux[0];
        int v = aux[1];
        int p = aux[2];

        if (visited[v]++) continue;
        T[v] = p;

        for (ii pr: G[v]){
            int dd = d + pr.second;
            int u = pr.first;

            Q.push({-dd, u, v});
        }
    }

    C = vi(n);
    for (int i = 0; i < n; ++i){
        for (ii p: G[i]){
            if (p.first == T[i]){
                C[i] = p.second;
            }
        }
    }
}

int rec(int v, int ch){
    if (v == n-1) return 0;

    int& ans = dp[v][ch];
    if (ans != -1) return ans;

    ans = C[v] + rec(T[v], ch);
    if (ch < k){
        for (ii p: G[v]){
            ans = max(ans, p.second + rec(p.first, ch+1));
        }
    }

    return ans;
}

int main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);

    int t;
    cin >> t;

    for (int ww = 0; ww < t; ++ww){
        cin >> n >> m >> k;

        G = vvii(n);
        T = vi(n);

        for (int i = 0; i < m; ++i){
```

```
            int x, y, w;
            cin >> x >> y >> w;
            --x; --y;

            G[x].push_back({y, w});
            G[y].push_back({x, w});
        }

        dijkstra();

        dp = vvi(n, vi(k+1, -1));
        cout << rec(0, 0) << '\n';
    }
}
```

Viernes, 8 de abril de 2022