2 Examen Algoritmia Manuel Morales

Q)

Cuando nosotros deseamos simular un poco la vista del grafo y cuando sabemos la cantidad exacta de elementos que tendra de conexiones y de nodos es cuando elejimos una matriz de adjacencia como en un caso de un input asi:

```
+++++++
+.....+
+.....+
+.....+
+++++++
```

Mientras que cuando la simulacion de este llega a hacer compleja e incluso podria decirse que muy laboriosa y solo nos interesa saber la informacion de que nodos con que nodos estan conectados es cuando usamos la lista de adjacencia es por ella que en esta es mas comun el uso de algun algoritmo de busqueda de la ruta mas corta, etc.

Q1)

Si precisamente es gracias a ello que en un map por ejemplo no puede aver keys repetidos, es por ello que si nosotros en java usamos el metodo equals tambien deberiamos revisar el metodo hascode para evitar estas colisiones.

Q2)

Preferiria el uso del hashing cuando no interesa el orden de los datos por que para el uso del linear o binary search es requerido que mis datos esten ordenados y para lograr esto necesitaria ocupar un tiempo de complejidad mas alto.

Q3)

La diferencia de user un map a un unordered_map en cpp es practicamente dependiente al problema, si es que en el problema nos pide obtener los datos de una manera ordenada nos convendria usar un map ya que en cada insercion el tiempo de complejidad seria log n, mientras que cuando nosotros queremos reordenar todo el map por nuestra cuenta esto minimamente nos pesaria un tiempo de complejidad n log n, es por ello que usamos map, pero en el caso que no nos interese el orden de los datos es cuando deberiamos usar por mucho el unordered_map ya que de esta manera la insercion la verificacion de un elemento si esta ahi, la eliminacion costarian un tiempo de complejidad 1.

Q4)

En el caso de c++ mi mejor opcion seria usar un vector, ya que en este podemos acceder a los mismo elementos del stack y en otro caso tambien utilizaria dequeue que es un stack pero que al mismo tiempo tambien llega a hacer un queue.

Q5)

```
#ifdef LOCAL
#include "/home/morven/Desktop/code/manuel-competitive/conf/debug.h"
#define line cerr << "-------------------" << endl;
#else
#define deb(x...)
#define line
#endif

#include <bits/stdc++.h>
using namespace std;

template <typename... T>
void cinn(T &...args)
{
  ((cin >> args), ...);
}

template <typename... T>
void coutt(const T &...args)
{
  __typeof(sizeof...(T)) i = 1;
  ((cout << args << (i++ != sizeof...(T) ? " " : "")), ...);
  cout << '\n';
}

template <typename T>
void couts(const T &xs)
{
  __typeof(xs.size()) i = 1;
  for (auto &x : xs)
  {
    cout << x << (i++ == xs.size() ? '\n' : ' ');
  }
}

#define endl '\n'
#define ll long long
#define F first
#define S second
#define pb push_back
#define sz size
#define all(x) begin(x), end(x)
#define sortt(x) sort(all(x))
#define each(x, xs) for (auto &x : (xs))
#define rep(i, be, en) for (__typeof(en) i = (be); i < (en); i++)
#define per(i, be) for (__typeof(be) i = (be)-1; i >= 0; i--)
#define strInt(str) stoi(str)
#define chrInt(chr) (int)chr - 48
#define ensp(i, n) (i == n - 1 ? '\n' : ' ')
#define readline(x) getline(cin, x)
```

```
const long long int INF = INT64_MAX;
const long long int MOD = 1e9 + 7;
const long double PI = acos(-1);
const vector<int> DX{1, 0, -1, 0}, DY{0, 1, 0, -1};

void _()
{
  ios::sync_with_stdio(0);
  cin.tie(0);
  cout.tie(0);
#ifdef LOCAL
  freopen("/home/morven/Desktop/code/manuel-competitive/conf/main.in", "r",
stdin);
  freopen("/home/morven/Desktop/code/manuel-competitive/conf/main.out",
"w", stdout);
  freopen("/home/morven/Desktop/code/manuel-competitive/conf/main.err",
"w", stderr);
#endif
}

void solve();

int main()
{
  _();
  int T;
  T = 1;
  // cin >> T;
  rep(t, 0, T) { solve(); }
  return 0;
}

const int MAXN = 202020;
int n, q, x;
map<int, int> mp;

/**
 * Time Complexity: O(log n)
 * Space Complexity: O(n)
 */
map<int, int> add(map<int, int> mp, int x)
{
  if (!mp[x])
  {
    mp[x]++;
    return mp;
  }
  return mp;
}

void solve()
{
  cinn(n);
```

```
    rep(i, 0, n)
    {
      cinn(q);
      mp[q]++;
    }
    cinn(x);
    mp = add(mp, x);
    deb(mp);
    each(e, mp)
    {
      while (e.S--)
      {
        cout << e.F << " ";
      }
    }
  }
```