Cognoms **Nom** **DNI**

Competència Transversal EDA

- *Aquest exercici s'ha de lliurar a través de les pràctiques del Racó abans del dia de l'examen final de teoria.*
- *Poseu el vostre nom complet i número de DNI.*
- *Contesteu a totes les qüestions en el propi full de l'enunciat.*
- *Quan doneu una referència a una font (llibre, revista, web, etc.), seguiu la norma "ISO-690 (author-date, English)". Podeu generar les referències en aquest format a:*
  `https://www.grafiati.com/en/blogs/iso-690-citation-generator/`
- *Quan doneu una URL, si us plau escriviu **clarament** i useu Google URL Shortener:*
  `https://shorturl.at`

Volem representar una col·lecció de (sub)conjunts $S_1, \ldots, S_k$ dels nombres $0, 1, 2, \ldots$
Els conjunts són disjunts dos a dos, és a dir $S_i \cap S_j = \emptyset$ si $i \neq j$.
Volem poder dur a terme les següents operacions:

- *make*$()$: si x és el primer natural que no apareix a la col·lecció, hi afegeix el conjunt $\{x\}$.

  Per exemple, si la col·lecció és $\{\{0,1,3\},\{2\}\}$, després de fer *make*$()$ la col·lecció és $\{\{0,1,3\},\{2\},\{4\}\}$.

- *merge*(x,y): donats x, y pertanyents a conjunts diferents S_x i S_y de la col·lecció, hi afegeix el conjunt $S_x \cup S_y$. Els conjunts S_x i S_y s'esborren de la col·lecció.

  Per exemple, si la col·lecció és $\{\{0,1,3\},\{2\},\{4,5\}\}$, després de fer *merge*$(3,4)$ la col·lecció és $\{\{0,1,3,4,5\},\{2\}\}$.

- *find*(x): troba el conjunt al qual pertany x.

  Cada conjunt de la col·lecció té un *representant* que l'identifica. El que fa la funció *find*(x) més concretament és retornar el representant del conjunt al qual pertany x. El representant pot ser qualsevol element del conjunt, però s'ha de complir que *find*(x) sempre retorna el mateix per a tots els elements x del mateix conjunt, si aquest no canvia (és a dir, si no es fan crides a *merge* que l'afectin).

  Per exemple, si la col·lecció és $\{\{1,3\},\{0,2\}\}$, llavors les crides *find*(1) i *find*(3) o bé retornen totes dues 1, o bé retornen totes dues 3.

Una forma d'implementar aquesta estructura de dades és mitjançant un bosc d'arbres arrelats. Cada arbre representa un conjunt, i la seva arrel és el representant del conjunt. Els nodes que l'arbre, cadascun dels quals apunta al seu pare (llevat de l'arrel), són els elements del conjunt.

Considereu el bosc d'arbres arrelats de la Figura 1:

Cognoms **Nom** **DNI**

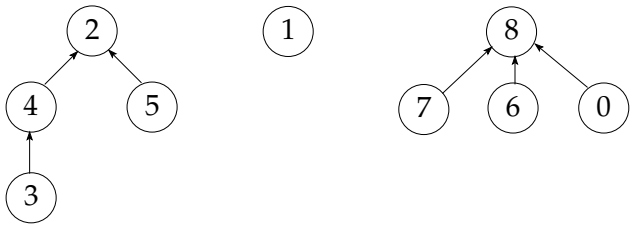

Figura 1: Bosc d'arbres arrelats.

(a) Quina és la col·lecció de conjunts disjunts representada pel bosc de la Figura 1?

{{2,3,4,5},{1},{0,6,7,8}}

(b) Escriviu el valor retornat en fer les següents crides sobre el bosc de la Figura 1:

- *find*(1): 1
- *find*(4): 2
- *find*(6): 8
- *find*(2): 2

(c) Un bosc d'arbres arrelats en què els nodes són els naturals $0, 1, \ldots, n-1$ es pot representar mitjançant un vector p de mida n en què $p[i]$ és -1 si i és arrel, o el pare del node i altrament.

Escriviu la representació en vector del bosc de la Figura 1:

i	0	1	2	3	4	5	6	7	8
$p[i]$	8	-1	-1	4	2	2	8	8	-1

(d) Implementeu les funcions *make* i *merge* de la següent classe de C++ usant la tècnica de l'unió per rang (*union by rank*):

Cognoms **Nom** **DNI**

```
class DisjointSetCollection {

  vector<int> p; // pare
  vector<int> r; // rang

public:
  void make();
  void merge(int x, int y);
  int find(int x);
};

void DisjointSetCollection::make() {
    int x = p.size();
    p.push_back(-1); // Creo nou conjunt
    r.push_back(0);   // de rang 0
}

void DisjointSetCollection::merge(int x, int y) {
    int rootX = find(x);
    int rootY = find(y);
    if (r[rootX] > r[rootY]) p[rootY] = rootX;        // Moc l'arbre petit
    else if (r[rootX] < r[rootY]) p[rootX] = rootY;  // com a branca del gran
    else {
       p[rootY] = rootX;
       r[rootX]++;
    }
}
```

(e) Implementeu la funció *find* usant la tècnica de compressió de camins (*path compression*):

```
int DisjointSetCollection::find(int x) {
    if (p[x] == -1) return x;
    else {
       p[x] = find(p[x]); // compressió de camins
       return p[x];
    }
}
```

(f) Doneu la referència de la font que heu consultat per a les tècniques de l'unió per rang i de la compressió de camins. Justifiqueu la vostra elecció.

Cognoms **Nom** **DNI**

Referència:

WILLIAMFISET. Union Find Path Compression [video]. YouTube. 8 April 2017 [viewed 24 May 2024]. Available from: https://shorturl.at/yOYlz

SHARMA, Harshit. Disjoint Set Unions by Rank and Path Compression. Medium [online]. 26 June 2020 [viewed 24 May 2024]. Available from: https://shorturl.at/2UluT

Justificació:

El vídeo de Williamfiset proporciona una explicació visual i detallada de com funcionen les tècniques de compressió de camins i unió per rang en les estructures de dades de conjunts disjunts. Els exemples pràctics i les animacions m'han ajudat a entendre millor els conceptes.

L'article de Harshit Sharma explica les tècniques de manera clara i concisa, amb codis d'exemple que demostren la implementació de les tècniques de l'unió per rang i compressió de camins. Mitjançant aquests codis he pogut assegurar la funcionalitat del codi.

(g) Dibuixeu el bosc obtingut després de cridar la funció $find(3)$ sobre el bosc de la Figura 1:

Nota: es pot demostrar que si s'usen les tècniques d'unió per rang i compressió de camins aleshores el cost de fer m operacions de *make*, *merge* i *find*, n de les quals són *make*, és $O(m\alpha(n))$. La funció $\alpha(n)$ és l'anomenada *funció inversa d'Ackermann* i val com a molt 4 per a tot n que pugui aparèixer a la pràctica (que forma que el cost és essencialment lineal).