

Integrantes:

- Tony Santiago Montes Buitrago - 202014562 [t.montes@uniandes.edu.co](mailto:t.montes@uniandes.edu.co)
- Isaac David Bermudez Lara - 202014146 [i.bermudezl@uniandes.edu.co](mailto:i.bermudezl@uniandes.edu.co)
- Valeria Pinzón Sierra - 202014948 [v.pinzon3@uniandes.edu.co](mailto:v.pinzon3@uniandes.edu.co)

Bono: requerimiento 6

Requerimiento 1

```
def buenasPelículas(director_name, casting, details) -> tuple:

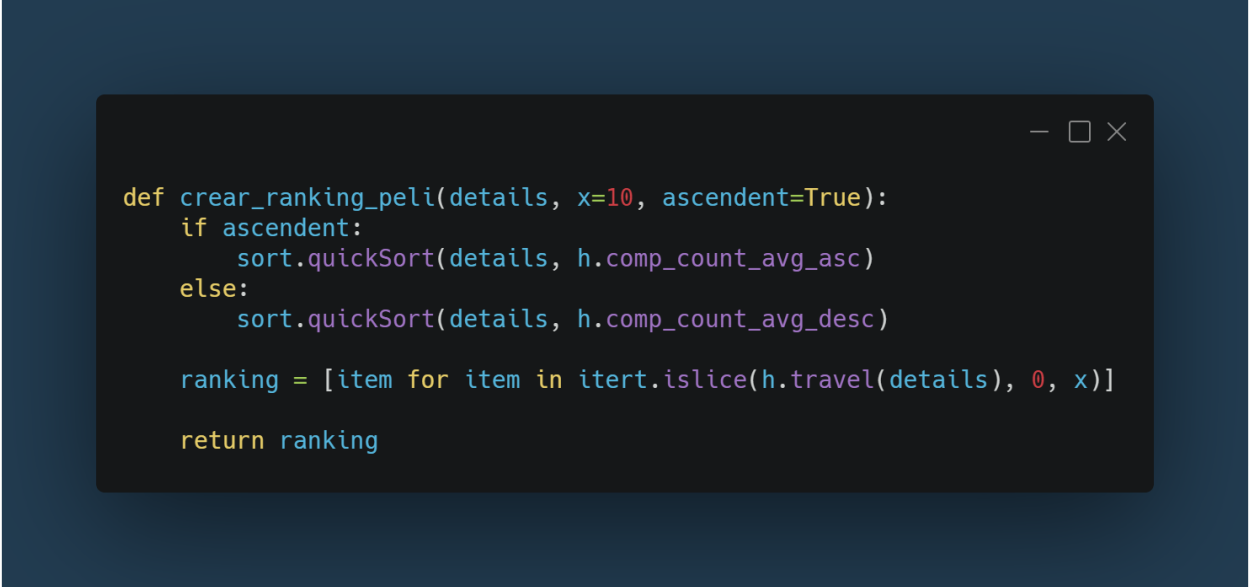
    lst1 = [
        node["id"]
        for node in h.travel(casting)
        if node["director_name"] == director_name
    ]

    lst2 = [
        float(node["vote_average"])
        for node in h.travel(details)
        if node["id"] in lst1 and float(node["vote_average"]) >= 6
    ]

    if len(lst2) <= 0:
        return (0, 0)
    else:
        return (len(lst2), (sum(lst2)) / len(lst2))
```

$O(n)$

## Requerimiento 2



```
def crear_ranking_peli(details, x=10, ascendent=True):  
    if ascendent:  
        sort.quickSort(details, h.comp_count_avg_asc)  
    else:  
        sort.quickSort(details, h.comp_count_avg_desc)  
  
    ranking = [item for item in itert.islice(h.travel(details), 0, x)]  
  
    return ranking
```

$O(n \log n)$

### Requerimiento 3

```
def conocer_director(details, casting, director_name) -> dict:

    lst = [
        node["id"]
        for node in h.travel(casting)
        if node["director_name"] == director_name
    ]

    info = []
    for node in h.travel(details):
        if node["id"] in lst:
            d = {
                "id": node["id"],
                "title": node["title"],
                "vote_average": node["vote_average"],
            }
            info.append(d)

    return info
```

$O(n)$

#### Requerimiento 4

```
def conocer_actor(details, casting, actorName) -> dict:
    # Este es el requerimiento opcional.
    lista = [
        node["id"]
        for node in h.travel(casting)
        if node["actor1_name"] == actorName
        or node["actor2_name"] == actorName
        or node["actor3_name"] == actorName
        or node["actor4_name"] == actorName
        or node["actor5_name"] == actorName
    ]
    actInf = []
    for node in h.travel(details):
        if node["id"] in lista:
            ac = {
                "num": {node["id"]},
                "avg": {node["vote_average"]},
                "title": node["title"],
                "director_name": node["director_name"],
            }
            ac["numPeliculas"] = len(ac["num"])
            ac["vote_average"] = ac["avg"] / len(ac["avg"])
            actInf.append(ac)

    return actInf
```

$O(n)$

## Requerimiento 5

```
def entender_genero(details, genero):  
    lista_genero = h.filter(details, "genres", genero, impl="ARRAY_LIST")  
    length = lt.size(lista_genero)  
  
    avg_vote_lst = [int(i["vote_count"]) for i in h.travel(lista_genero)]  
    avg_vote = sum(avg_vote_lst) / len(avg_vote_lst)  
  
    return lista_genero, length, avg_vote
```

$O(n)$

## Requerimiento 6

```
def crear_ranking_genero(details, genero, retrieve=10, ascendent=True):  
    dgend = h.filter(details, "genres", genero, impl="ARRAY_LIST")  
    if ascendent:  
        sort.quickSort(dgend, h.comp_count_avg_asc)  
    else:  
        sort.quickSort(dgend, h.comp_count_avg_desc)  
    raw = [item for item in itert.islice(h.travel(dgend), 0, retrieve)]  
    ranking = [peli["title"] for peli in raw]  
    avg_vote_lst = [int(i["vote_count"]) for i in raw]  
    avg_vote = sum(avg_vote_lst) / len(avg_vote_lst)  
    return ranking, avg_vote
```

$O(n \log n)$