Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2133 — Estructuras de Datos y Algoritmos
2023 - 1

# Tarea 0

**Fecha de entrega código:** 27 de Marzo del 2023
**Link a generador de repos:** Ir a github

## Objetivos

- Comprender las diferencias entre Arrays y Listas ligadas
- Familiarizarse con el uso de punteros y manejo de memoria
- Aplicar algoritmos de sorting y tecnicas de implementación

## Introducción

Feliz de que te tomaste un tremendo helado para capear el calor, vas caminando con todo el entusiasmo hacia tu clase (favorita podrían decir algunos, cuestionable otros) de Estructuras de Datos y Algoritmos, cuando te sorprendes al ver que afuera de la sala los profes están discutiendo. El profe Droguett sostiene que el auto favorito de la población chilena es Francesco Virgolini, representante de los ¨Totte chile x los choripanes gratis¨. Por su parte, el profe Bugedo no puede tolerar semejante mentira sosteniendo que el verdadero auto en el corazón de Chile es El Rayo Mcqueen, representante de ¨A 4 ruedas x los completos felices¨.

Lamentablemente, dada la relevancia de la discusión, los profes deciden dejar de ser besties hasta resolver el asunto y deciden someter esto a votación con sus estudiantes, incluyendo a todos los personajes del universo mecánico-ideológico de CARS. Además, para asegurar imparcialidad, te piden a ti que lleves a cabo la votación a través de la creación de un código que pueda registrar y contabilizar votos, así como también mostrar a los ganadores bajo diferentes sistemas de votación.

**Lograrás que los profes vuelvan a ser "amiguis"?** Les ayudantes decimos que obvio que sí!

FIUUUUUUM

Cuchau

# Problema

En un principio, debes procesar los votos que te entregaremos y generar ciertos conteos de ellos, para luego generar simulaciones bajo dos sistemas de votación diferentes y manejar ciertos eventos en cada caso. Es importante que al llegar a la Parte C pongas atención en el sistema de votación pedido pues posee ciertas interesantes particularidades en su funcionamiento.

### Candidatos

Las tendencias de los candidatos están determinadas en un espectro. Si en la simulación existen $N$ `Candidate`, numerados de 0 a $N-1$, el primer candidato será el menos a favor y el último será el más radical a favor de X.

Definiremos la distancia ideológica $d(i, j)$ entre un candidato $i$ y otro $j$ como

$$d(i, j) = min(N - |i - j|, |i - j|)$$

En otras palabras, es la distancia entre sus posiciones en el espectro, considerando que para este caso la *teoría de la herradura*[1] es cierta. La idea principal es imaginar que los candidatos están ordenados de tal forma que forman un círculo, de manera que la distancia entre el primero y el último sea 1.

### Votantes

- Cada `Voter` posee un `ID` y un `VOTE` que es un número entre 0 y $N - 1$.
- No se sabe el número de votos que cada candidato tendrá apriori
- Hay votos que pueden ser *anulados*

### (Solo Parte C) Paso de votos

En la parte C se simulará el sistema $STV$[2] ajustado para el contexto del desafío. Donde los votos de los perdedores y parte de los votos de los ganadores parciales serán transferidos según la siguientes regla:

Dado un `Candidate` $i$ que debe transferir $V$ votos, se le transferirán los votos al `Candidate` $j$ tal que $d(i, j)$ sea mínima. En caso de que exista empate entre dos, se le transferirán los votos solo al `Candidate` con menor `ID`.

Ejemplo:

- 4 Candidates, $[C_0, C_1, C_2, C_3]$. $C_2$ debe transferir votos.
- Tanto $C_1$ como $C_3$ poseen la misma distancia 1 hacia $C_2$
- $C_1.id < C_3.id$ por lo tanto los votos se transfieren a $C_1$

[1] La distancia entre extremos del espectro es minima
[2] Single Transferable Vote

# Eventos

Para facilitar la implementación, los eventos estarán divididos en tres categorías:

## Parte A. Procesamiento de votos (30 %)

Esta etapa contempla todo lo que es el registro y procesamiento de los votos en una elección. Se tiene una cantidad fija de $N$ candidatos. Y los votos van llegando como eventos

### 1. `REGISTRAR-VOTO voterID CandidateID`

El evento registra un nuevo votante y su voto hacia un candidato en específico

```
REGISTRAR-VOTO voterID CandidateID
```

Se ha de imprimir el ID de la persona agregada en el siguiente formato **cada vez** que se registra un nuevo voto:

```
VOTO REGISTRADO voterID
```

### 2. `CONTEO-PARCIAL`

El evento muestra los votos que han sido registrados hasta el momento seguido de un conteo de este total parcial agrupado por ID de candidato, lo cual se ha de imprimir en el siguiente formato:

```
CONTEO-PARCIAL
    CANDIDATE 1
        VOTE voterID
        ...
    CANDIDATE 2
    ...
TOTAL PARCIAL DE VOTOS: 3
```

En caso de no haberse registrado ningún voto hasta el momento, se deberá imprimir lo siguiente:

```
CONTEO-PARCIAL
    CANDIDATE 1
        NO HAY VOTOS REGISTRADOS
    CANDIDATE 2
        ...
    ...
TOTAL PARCIAL DE VOTOS: 0
```

### 3. `CONTEO-TOTAL`

Este evento entregará los resultados para cada candidato. (Similar al punto 2. Pero con el rango completo)

```
CONTEO-TOTAL
```

```
CONTEO-TOTAL
    CANDIDATO 1: 200
    CANDIDATO 2: 25
    CANDIDATO 3: 741
    CANDIDATO 4: 122
    ...
TOTAL VOTOS: 101332
```

## B. Resultados por mayoría absoluta (30 %)

Ahora que ya se han contabilizado todos los votos, quieres saber quiénes son los ganadores de esta elección. Primero, piensas en elegir a los candidatos con la mayor cantidad de votos. Para eso, requieres de crear las siguientes funciones.

### 1. `CONTEO-RANGO minVotes maxVotes`

Con este evento se podrá ver los candidatos cuya cantidad de votos se encuentre entre un rango determinado.

```
CONTEO-RANGO 80 800
```

Al procesar la línea anterior, se debe imprimir la cantidad de votos de los candidatos que han recibido entre 80 y 800 votos (inclusive).

Por ejemplo, si existen 3 candidatos con 100, 500 y 2000 votos respectivamente. Y la consulta es CONTEO-RANGO 80 800. Solo se entregarán los candidatos 0 y 1.

```
CONTEO-RANGO 80 800
    CANDIDATO 1: 100
    CANDIDATO 2: 500
TOTAL DE VOTOS RANGO: 600
```

### 2. `ORDENAR-CANDIDATOS`

Esta acción se dedica a **IMPRIMIR** los candidatos por orden de mayor a menor votos recibidos como primera opción.

Al terminar de ordenar a los candidatos, imprime el orden obtenido, por ejemplo:

```
CANDIDATOS-ORDENADOS
    CANDIDATO 4 7203
    CANDIDATO 2 5821
    CANDIDATO 3 2938
    CANDIDATO 0 299
    CANDIDATO 5 0
TOTAL DE VOTOS: 16261
```

OJO: El comportamiento en el resto de las operaciones no se debería ver afectado por esta operación.

### 3. `ANULAR-VOTO voterID candidateID`

Lamentablemente, hay votos inválidos que deben ser anulados para el correcto funcionamiento de las elecciones. Este evento indica que el voto de id `voterID` y por el candidato `candidateID` debe ser eliminado. Importante: Esta operación debe mantener el resto de la simulación consistente. (En caso de nuevos eventos, los resultados deben variar)

```
ANULAR-VOTO 1451 4
```

En caso de existir el voto se entrega

```
VOTO ELIMINADO CORRECTAMENTE
```

En caso de no existir un voto para tal candidato con el ID

```
NO SE ENCONTRO UN VOTO VALIDO CON ID voterID
```

## C. Resultados por STV (40 %)

*Esta sección considera la regla de traspaso de votos mencionada en la sección de Problema.*

STV es un sistema electoral que permite tener múltiples ganadores basándose en un voto con preferencias. Puedes ver este video para más información. Sin embargo, para simplificar la modelación, los votantes solo votan por un candidato y luego el traspaso de votos se realiza según cercanía hacia otros candidatos.

Importante: Estos eventos sólo ocurrirán después de terminada la emisión de todos los votos.

### 1. `ELIMINAR-CANDIDATO`

La operación elimina al candidato con **menos votos** y **todos** sus votos son transferidos al candidato más próximo según las reglas mencionadas anteriormente. En caso de un **empate** (2 o más candidatos presentan la misma cantidad de votos que resulta ser la menor), se debe eliminar al candidato con el menor id `candidateID`. El output deberá ser de la siguiente forma:

```
CANDIDATO A HA SIDO ELIMINADO
CANDIDATO B HA RECIBIDO 299 VOTOS
```

Notar que una vez eliminado un candidato, se dejara de considerar para próximos traspasos de votos. Sin embargo, deberá seguir apareciendo en el conteo de votos con su nueva cantidad de votos, es decir, cero.

### 2. `TRASPASAR-EXCESO-VOTOS CANDIDATOA M`

Este paso definirá como ganador al candidatoA. Además, en caso de que el candidato posea una cantidad de votos $V > M$, los últimos $V - M$ votos $[M + 1, \cdots, V]$ serán transferidos al candidato más próximo en el espectro que siga en competencia.

```
{{ NOMBRE CANDIDATO A}} HA SIDO ELEGIDO
{{ NOMBRE CANDIDATO B}} HA RECIBIDO 299 VOTOS
```

Por ejemplo, si el candidato A posee 30 votos y M es 25. Deberá transferir sus **últimos** $30 - 25 = 5$ votos hacia el candidato que se encuentre más cercano en el espectro. Se debe actualizar la nueva cantidad de votos de cada uno de los candidatos para un próximo conteo de votos.

# Ejecución

Tu programa se debe compilar con el comando `make` y debe generar un ejecutable de nombre `dccars` que se ejecuta con el siguiente comando:

```
./dccars input.txt output.txt
```

Donde input será un archivo con los eventos a simular y output el archivo a guardar los resultados.
Tu tarea será ejecutada con archivos de creciente dificultad, asignando puntaje a cada una de estas ejecuciones que tenga un output igual al esperado. A continuación detallaremos los archivos de input y output.

### Input

El archivo de input comenzará con el número de `Candidates N`, la siguiente línea sería un número `E` que corresponde al número de eventos a recibir. (Esto incluye emisiones de votos, consultas, operaciones.
Se verían con el siguiente formato:

```
N
E
REGISTRAR-VOTO 0 2
CONTEO-TOTAL
REGISTRAR-VOTO 1 3
REGISTRAR-VOTO 2 0
REGISTRAR-VOTO 3 10
ORDENAR-CANDIDATOS
...
```

En este input describe `N` candidatos, y `E` eventos en total.

### Output

Tu output debe consistir de un archivo con la información solicitada por cada evento en el archivo de input.

## Evaluación

La nota de tu tarea es calculada a partir de testcases e Input/Output. Y se descompone de la siguiente forma

- 90 % a la nota entre las partes A, B y C
  - 30 % a la nota de los tests de la parte A
  - 35 % a la nota de los tests de la parte B
  - 35 % a la nota de los tests de la parte C
- 10 % Por manejo de memoria
  - 5 % Por no poseer leaks de memoria
  - 5 % Por no poseer errores de memoria

Para cada test de evaluación, tu programa deberá entregar el output correcto en **menos de 10 segundos** y utilizar menos de 1 GB de RAM[3]. De lo contrario, recibirás 0 puntos en ese test.

## Recomendación de tus ayudantes

Las tareas requieren de mucha dedicación de tiempo generalmente, por lo que desde ya te recomendamos distribuir tu tiempo considerando los plazos definidos. Así mismo, te recomendamos fuertemente que antes de empezar a programar tu tarea, leas el enunciado y te dediques a entender de manera profunda lo que te pedimos. Una vez que hayas comprendido el enunciado, dedica el tiempo que sea necesario para la planificación y modelación de tu solución, para posteriormente poder programar de manera eficiente. Estos son consejos de tus ayudantes que te pueden ayudar a pasar el ramo :)

## Entrega

**Código:** GIT - Rama master del repositorio asignado. Se entrega a más tardar el día de entrega a las 23:59 hora de Chile continental.

**Atraso:** Esta tarea no considera cupones ni política de atrasos, cualquier atraso debe ser justificado con el equipo docente del curso.

## Integridad académica

Este curso se adscribe al Código de Honor establecido por la Escuela de Ingeniería. Todo trabajo evaluado en este curso debe ser hecho **individualmente** por el alumno y **sin apoyo de terceros**. Se espera que los alumnos mantengan altos estándares de honestidad académica, acorde al Código de Honor de la Universidad. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un Procedimiento Sumario. Es responsabilidad de cada alumno conocer y respetar el documento sobre Integridad Académica publicado por la Dirección de Pregrado de la Escuela de Ingeniería.

[3] Puedes revisarlo con el comando htop