CS-240: Programming Assignment Phase 1

Winter Semester 2024-2025

John Malliotakis – jmal@csd.uoc.gr

25/10/2024

Project Concept

The idea: loosely simulate greek elections

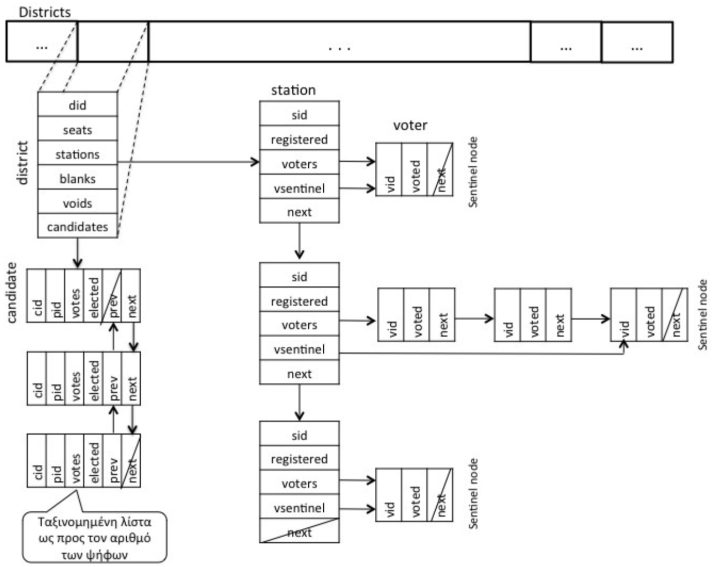
- **5** *Parties* (*candidate* lists)
- **56** *Districts*
  - Party *candidates* registered per district
  - Election *stations* per district
    - *Voters* registered per station
- Final formed *parliament* → Elected candidates from all parties

How the project works:

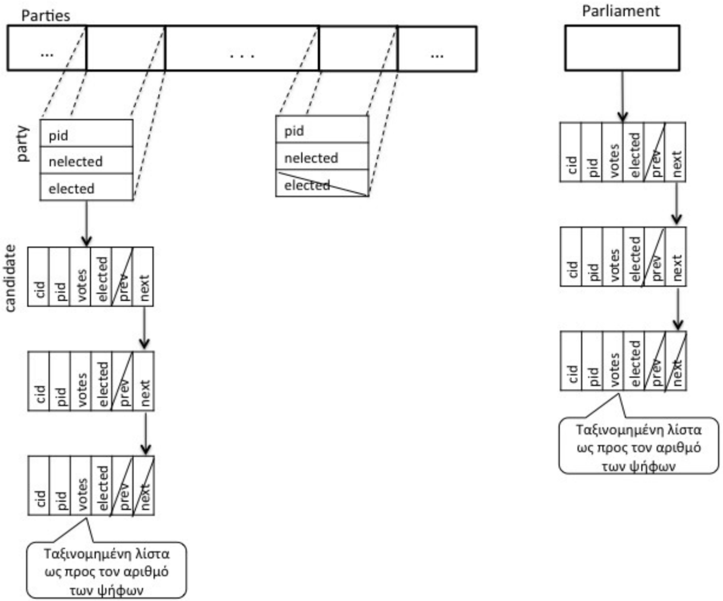
- Input testfile *parameter*
- Testfile contains "*events*" → 1 per line
- Events → *Actions* on parties/districts/stations/voters/candidates

Structures & Organization

Overview

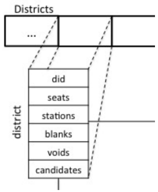

Districts
...
. . .
...
...
district
did
seats
stations
blanks
voids
candidates
station
sid
registered
voters
vsentinel
next
voter
vid
voted
next
Sentinel node
candidate
cid
pid
votes
elected
prev
next
Ταξινομημένη λίστα ως προς τον αριθμό των ψήφων

Overview


Parties
...
. . .
...
party
pid
nelected
elected
candidate
cid
pid
votes
elected
prev
next
Ταξινομημένη λίστα ως προς τον αριθμό των ψήφων
Parliament
Ταξινομημένη λίστα ως προς τον αριθμό των ψήφων

Districts

Stored in global districts array

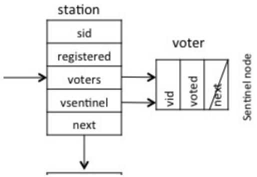


Fields:

did	unique district ID
seats	total seats to be distributed
allotted	seats distributed after first vote count (event M)
blanks	blank (i.e., no candidate selected) votes
voids	void (i.e., invalid) votes
stations	unsorted, singly-linked list of district voting stations
candidates	sorted ($\downarrow$ votes), doubly-linked district candidate list

Voting Stations

Stored in district stations list



Fields:

sid	unique station ID
registered	total registered voters
voters	unsorted, singly-linked registered voter list with sentinel node
vsentinel	pointer to registered voter list sentinel node
next	next station pointer

Candidates

Stored in district, party, and parliament lists

- Always sorted, based on decreasing vote count
- Doubly-linked in **districts**, singly-linked in parties, parliament $\rightarrow$ prev field unused

Fields:

cid	unique candidate ID $\neq$ 0 or 1 $\rightarrow$ reserved for blanks and voids
pid	party ID, to which candidate belongs
votes	total votes received, used for sorting
elected	boolean (0 or 1) $\rightarrow$ was this candidate elected?
prev	pointer to previous candidate in district only
next	pointer to next candidate in district/party/parliament

Parties & Parliament

Both store singly-linked candidate lists

- Party $\rightarrow$ candidates belonging to the specific party
- Parliament $\rightarrow$ elected candidates from multiple parties

Party fields:

pid	unique party ID
nelected	number of elected candidates from party
elected	Party elected candidate list

Parliament fields:

members	List of candidates elected to parliament

Events

A - Announce Elections

Initialize global structures

- Integer fields initialized to **-1**
- Pointer fields initialized to **NULL**

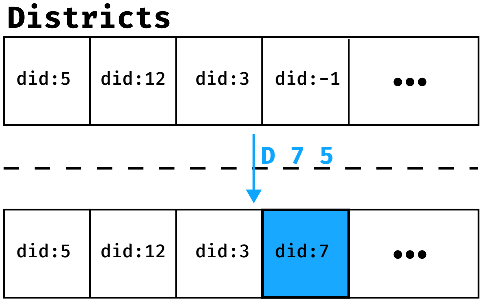
Actions:

- Initialize districts array (56 uninitialized district structures)
- Initialize parties array (5 unitialized parties)
- Initialize parliament structure instance (empty elected candidate list)

D <did> <seats> - Create District

Create a new election district

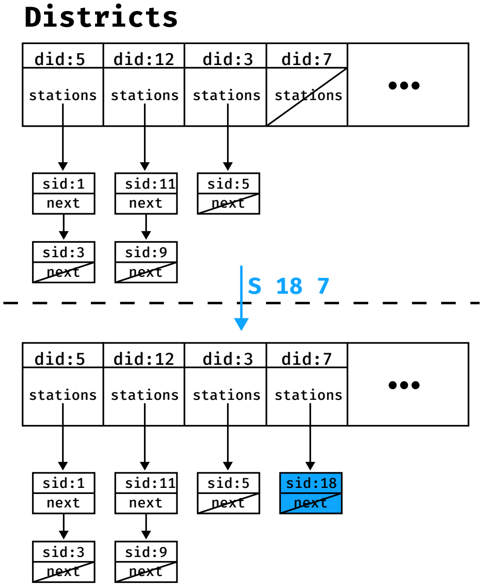
- Initialize with ID <did>, total seats <seats>
- Empty station, candidate lists
- Place in first empty slot of districts array
  - $O(1)$ time complexity
  - Requires extra variables



S <sid> <did> - Create Station

Create a new voting station

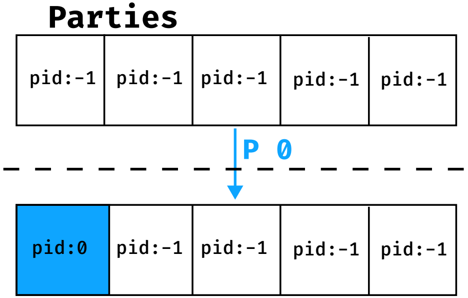
- Initialize a new station → 0 registered voters, empty voter list (sentinel node only)
- Find district with ID <did> in districts array
- Add a new station with ID <sid> to district stations list



P <pid> - Create Party

Like event D for districts

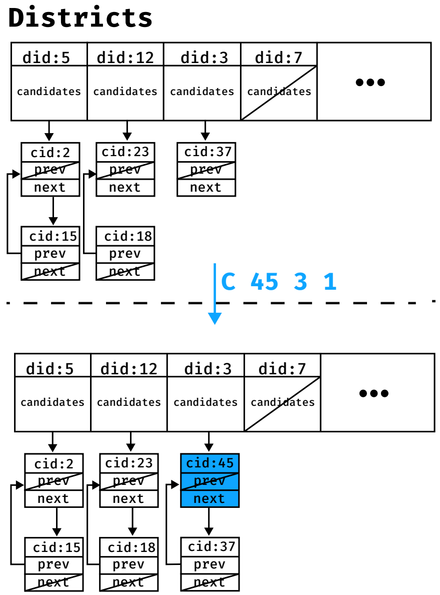
- Use first empty slot of parties array
- Initialize the slot with ID <pid>, empty candidate list



C <cid> <did> <pid> - Register Candidate

Register a new candidate

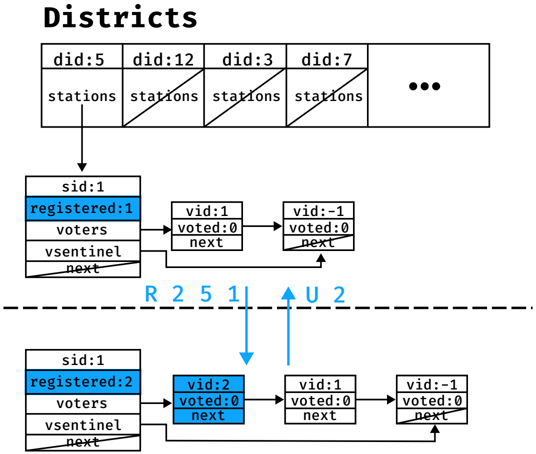
- Initialize ID with <cid>, party ID with <pid>
- Locate district ID <did> in districts array
- Insert candidate to district candidate list (and not party list)



R <vid> <did> <sid> - Register voter

Register a new voter

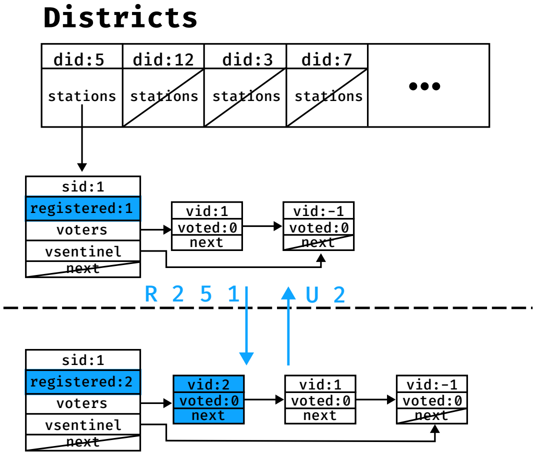
- Initialize with voter ID <vid>
- Locate district with ID <did>
- Add voter to station <sid> voter list
- Increment station <sid> registered voters count



U <vid> - Unregister voter

Remove a registered voter

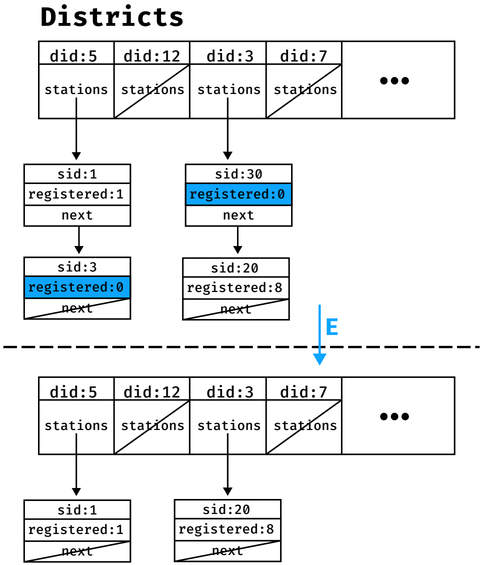
- Opposite process to R
- Decrement station registered voters count



E - Delete Empty Stations

Remove voting stations with 0 registered voters

- Iterate districts array
- For each district:
    - Iterate district voting stations list
    - Check registered field
    - If $= 0 \rightarrow$ Remove from list
- Time complexity: $O(n)$ ($n = \#$ voting stations)



V <vid> <sid> <cid> - Vote

Cast a vote:

1. Search district station lists for station <sid>
2. Locate voter <vid> in station <sid> registered voter list
3. Update voter <vid> **elected** field: $0 \rightarrow 1$
4. Check <cid>
   - If 0 $\longrightarrow$ `district`→**blanks** = `district`→**blanks** + 1
   - If 1 $\longrightarrow$ `district`→**voids** = `district`→**voids** + 1
5. If <cid> is valid candidate $\longrightarrow$ <cid> in district candidate list
6. `candidate`→**votes** = `candidate`→**votes** + 1
7. If `candidate`→**prev**→**votes** < `candidate`→**votes**:
   - Must maintain decreasing vote sort!
   - Swap `candidate`→**prev**, `candidate`

M <did> - Count votes

Count votes for district <did>

1. Locate district <did> in district array
2. Initialize helper array with total valid votes per party
3. Iterate candidate list (1st iteration)
   - Add candidate **votes** to helper array slot (based on candidate **party ID**)
4. $\texttt{EklogikoMetro} = \frac{\text{Total valid votes (all parties)}}{\text{Total district seats}}$
5. For each party: $\texttt{ElectedSeats} = \lfloor \frac{\text{Total party votes}}{\texttt{EklogikoMetro}} \rfloor$
6. Store `ElectedSeats` per party in helper variables/array
7. Iterate candidate list (2nd and last iteration)
   - For each party **pid**, the first `ElectedSeats`[**pid**] candidates of the party are elected.
   - Change candidate **elected** field to 1
   - Copy candidate node to party elected candidate list
     - Sorted list insert!
   - Add 1 to party **nelected** field.
   - Add 1 to district **allotted** field.

G - Form Government

Form government by distributing leftover district seats

1. Find party ID with the most total elected candidates (**nelected**)
2. For each district:
3. Calculate leftover seats = **seats - allotted**
4. Iterate district candidate list:
5. First (simple) case: Party with most total seats gets leftover seats
   - Elect remaining unelected party candidates as in event M
6. Second (difficult) case: Party with most seats does not have enough unelected candidates for leftover seats
   - Elect as many candidates as possible from party with most total seats
   - Distribute remaining leftover seats to unelected candidates based on total votes, regardless of party ID

N - Form Parliament

Form parliament from party elected candidate lists

- Merge party elected candidate lists
- Final list must also be sorted by decreasing votes!
- $O(n)$ time complexity, where n = total elected candidates (=300)

Some tips for a smooth project

Coding-related

- Split complex functions into smaller parts → Avoids errors, helps understanding
- Comment your code!
    - Helps both you and us understand the code
- Use `gdb` for quick debugging
    - Great tool to detect segfaults
    - Check tutorial on course website (`https://www.csd.uoc.gr/~hy240/current/material/assistiveClasses/gdb_tutorial.pdf`)
    - (**Optional**) `valgrind` for memory leaks

Some tips for a smooth project

Logistics-related

- *Divide and conquer*
  - Work on events one by one
  - If you're stuck on something, try something else and return
- *Ask* questions!
  - Utilize both mailing list and office hours
  - How can you learn if you do not ask?