Este é o CS50

Introdução à Ciência da Computação (CS50)

OpenCourseWare

Doar (https://cs50.harvard.edu/donate)

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan)
(https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/)
(https://www.reddit.com/user/davidjmalan)
(https://www.threads.net/@davidjmalan) (https://twitter.com/davidjmalan)

Escoamento

Problema a resolver

Você já conhece as eleições por maioria simples, que seguem um algoritmo muito simples para determinar o vencedor: cada eleitor tem direito a um voto, e o candidato com o maior número de votos vence.

Mas o sistema de votação por maioria simples tem algumas desvantagens. O que acontece, por exemplo, em uma eleição com três candidatos, e os votos abaixo são computados?

Ballot	Ballot	Ballot	Ballot	Ballot
Alice	Alice	Bob	Bob	Charlie

Uma votação por maioria simples declararia, neste caso, um empate entre Alice e Bob, já que cada um tem dois votos. Mas será esse o resultado correto?

Existe outro tipo de sistema de votação conhecido como sistema de votação por ordem de preferência. Nesse sistema, os eleitores podem votar em mais de um candidato. Em vez de votar

apenas em sua primeira opção, eles podem classificar os candidatos por ordem de preferência. As cédulas resultantes podem, portanto, ter a aparência abaixo.

Ballot	Ballot	Ballot	Ballot	Ballot
1. Alice	1. Alice	1. Bob	1. Bob	1. Charlie
2. Bob	2. Charlie	2. Alice	2. Alice	2. Alice
3. Charlie	3. Bob	3. Charlie	3. Charlie	3. Bob

Aqui, cada eleitor, além de especificar seu candidato de primeira preferência, também indicou sua segunda e terceira opções. E agora, o que antes era uma eleição empatada pode ter um vencedor. A disputa estava originalmente empatada entre Alice e Bob, então Charlie estava fora da corrida. Mas o eleitor que escolheu Charlie preferiu Alice a Bob, então Alice pode ser declarada a vencedora.

O sistema de votação por ordem de preferência também pode resolver outra possível desvantagem da votação por maioria simples. Veja as cédulas a seguir.

Ballot	Ballot	Ballot	Ballot	Ballot
1. Alice	1. Alice	1. Bob	1. Bob	1. Bob
2. Bob	2. Bob	2. Alice	2. Alice	2. Alice
3. Charlie	3. Charlie	3. Charlie	3. Charlie	3. Charlie

Ballot	Ballot	Ballot	Ballot
1. Charlie	1. Charlie	1. Charlie	1. Charlie
2. Alice	2. Alice	2. Bob	2. Bob
3. Bob	3. Bob	3. Alice	3. Alice

Quem deveria ganhar esta eleição? Em um sistema de votação por maioria simples, onde cada eleitor escolhe apenas sua primeira preferência, Charlie vence com quatro votos, contra apenas três de Bob e dois de Alice. Mas a maioria dos eleitores (5 de 9) preferiria Alice ou Bob em vez de Charlie. Ao considerar as preferências classificadas, um sistema de votação pode ser capaz de escolher um vencedor que reflita melhor as preferências dos eleitores.

Um exemplo de sistema de votação por ordem de preferência é o sistema de votação por eliminação instantânea. Em uma eleição por eliminação instantânea, os eleitores podem

classificar quantos candidatos desejarem. Se algum candidato obtiver a maioria (mais de 50%) dos votos de primeira preferência, esse candidato é declarado vencedor da eleição.

Se nenhum candidato obtiver mais de 50% dos votos, ocorre um "segundo turno instantâneo". O candidato que recebeu o menor número de votos é eliminado da eleição, e a segunda opção de quem o escolheu como primeira escolha passa a ser considerada. Por que fazer assim? Basicamente, isso simula o que teria acontecido se o candidato menos votado não tivesse participado da eleição.

O processo se repete: se nenhum candidato obtiver a maioria dos votos, o candidato com menos votos é eliminado, e todos que votaram nele passam a votar em sua próxima preferência (que ainda não tenha sido eliminada). Assim que um candidato obtiver a maioria, ele é declarado o vencedor.

Parece um pouco mais complicado do que uma votação por maioria simples, não é? Mas, sem dúvida, tem a vantagem de ser um sistema eleitoral onde o vencedor representa com mais precisão as preferências dos eleitores. Em um arquivo chamado `<nome_do_arquivo>` `runoff.c` dentro de uma pasta chamada `<nome_da_pasta>` `runoff`, crie um programa para simular um segundo turno.

Demonstração

```
Rank 1: Alice
Rank 2: Bob
Rank 3: Charlie

Alice
$ ./runoff Alice Bob Charlie
Number of voters: 3
Rank 1: David
Invalid vote.
$ ./runoff
Usage: runoff [candidate ...]
$
```

Recorded with **asciinema**

Código de Distribuição

▼ **Baixe o código de distribuição.**

Faça login em cs50.dev (https://cs50.dev/) , clique na janela do terminal e execute `cd` o comando. Você verá que o prompt do terminal será semelhante ao seguinte:

```
$
```

Em seguida, execute

```
wget https://cdn.cs50.net/2024/fall/psets/3/runoff.zip
```

para baixar um arquivo ZIP chamado `runoff.zip` para o seu espaço de código.

Em seguida, execute

```
unzip runoff.zip
```

para criar uma pasta chamada `runoff`. Você não precisa mais do arquivo ZIP, então pode executar

```
rm runoff.zip
```

e responda com "y" seguido de Enter quando solicitado para remover o arquivo ZIP que você baixou.

Agora digite

```
cd runoff
```

Em seguida, pressione Enter para entrar (ou seja, abrir) esse diretório. Seu prompt agora deve ser semelhante ao abaixo.

```
runoff/ $
```

Se tudo correr bem, você deverá executar

```
ls
```

e veja um arquivo chamado `runoff.c`. A execução `code runoff.c` deve abrir o arquivo onde você digitará o código para este conjunto de problemas. Caso contrário, refaça seus passos e veja se consegue determinar onde errou!

▼ **Entenda o código em`runoff.c`**

Sempre que você for ampliar a funcionalidade de um código existente, é melhor ter certeza de que primeiro o entende em seu estado atual.

Observe o início da `runoff.c` primeira linha. Duas constantes são definidas: uma `MAX_CANDIDATES` para o número máximo de candidatos na eleição e outra `MAX_VOTERS` para o número máximo de eleitores na eleição.

```
// Max voters and candidates
#define MAX_VOTERS 100
#define MAX_CANDIDATES 9
```

Observe que `MAX_CANDIDATES` é usado para dimensionar uma matriz `candidates`.

```
// Array of candidates
candidate candidates[MAX_CANDIDATES];
```

`candidates` é uma matriz de `candidate` s. Em `runoff.c` um , `candidate` há um `struct`. Cada `candidate` tem um `string` campo para seu `name`, um `int` representando o número de que `votes` eles têm atualmente e um `bool` valor chamado `eliminated` que indica se o candidato foi eliminado da eleição. A matriz `candidates` manterá o controle de todos os candidatos na eleição.

```
// Candidates have name, vote count, eliminated status
typedef struct
{
    string name;
    int votes;
    bool eliminated;
}
candidate;
```

Agora você pode entender melhor `preferences` a matriz bidimensional. A matriz `preferences[i]` representará todas as preferências do eleitor número `i`. O inteiro, `preferences[i][j]`, armazenará o índice do candidato, na `candidates` matriz, que é a `j`-ésima preferência do eleitor `i`.

```
// preferences[i][j] is jth preference for voter i
int preferences[MAX_VOTERS][MAX_CANDIDATES];
```

O programa também possui duas variáveis globais: `voter_count` e `candidate_count`.

```
// Numbers of voters and candidates
int voter_count;
int candidate_count;
```

Agora `main`, observe que, após determinar o número de candidatos e o número de eleitores, o loop principal de votação começa, dando a cada eleitor a oportunidade de votar. À medida que o eleitor insere suas preferências, a `vote` função é chamada para manter o controle de todas as preferências. Se, em algum momento, o voto for considerado inválido, o programa é encerrado.

Assim que todos os votos forem computados, inicia-se outro ciclo: este repetirá o processo de segundo turno, verificando se há um vencedor e eliminando o candidato com menos votos até que haja um vencedor.

A primeira chamada aqui é para uma função chamada `preferências` `tabulate`, que deve analisar as preferências de todos os eleitores e calcular o total de votos atual, observando o candidato de primeira escolha de cada eleitor que ainda não foi eliminado. Em seguida, a `print_winner` função deve imprimir o vencedor, se houver; se houver, o programa termina. Caso contrário, o programa precisa determinar o menor número de votos recebido por qualquer candidato ainda na disputa (por meio de uma chamada à função `menor número de votos` `find_min`). Se todos os candidatos estiverem empatados com o mesmo número de votos (conforme determinado pela `is_tie` função `empate`), a eleição é declarada empatada; caso contrário, o(s) candidato(s) com o menor número de votos é(são) eliminado(s) da eleição por meio de uma chamada à `eliminate` função `elimin`.

Se você olhar um pouco mais abaixo no arquivo, verá que o restante das funções `vote` — ` `tabulate` runoff.c`, `runoff.d` `print_winner`, ` `find_min` runoff.c`, `is_tie` `runoff.d` e `eliminate` `runoff.c` — fica a seu critério! **Você só deve modificar essas funções em `runoff.c`, embora possa adicionar `#include` arquivos de cabeçalho extras acima de `runoff.c`, se desejar.**

Dicas

Clique nos botões abaixo para ler algumas dicas!

▼ **Complete a `vote`função**

Conclua a `vote` função.

- A função recebe três argumentos: `voter`, `rank`, e `name`.
- Se `name` o nome corresponder ao de um candidato válido, você deverá atualizar a matriz de preferências globais para indicar que o eleitor `voter` tem esse candidato como sua `rank` preferência. Lembre-se de `0` que `1` é a primeira preferência, ` `1` 2` é a segunda preferência, e assim por diante. Você pode assumir que não haverá dois candidatos com o mesmo nome.
- Se a preferência for registrada com sucesso, a função deverá retornar ` true` `true`. Caso contrário, a função deverá retornar `false` `false`. Considere, por exemplo, quando ` `name` is` não é o nome de um dos candidatos.

Ao escrever seu código, considere estas dicas:

- Lembre-se que `candidate_count` armazena o número de candidatos na eleição.
- Lembre-se de que você pode usar `compare` `strcmp` (https://man.cs50.io/3/strcmp)para comparar duas strings.

- Lembre-se que `preferences[i][j]` armazena o índice do candidato que é a `j` preferência classificada do `i` eleitor.

▼ **Complete a `tabulate`função**

Conclua a `tabulate` função.

- A função deve atualizar o número de votos que `votes` cada candidato possui nesta fase do segundo turno.
- Lembre-se de que, em cada etapa do segundo turno, cada eleitor vota efetivamente em seu candidato preferido que ainda não tenha sido eliminado.

Ao escrever seu código, considere estas dicas:

- Lembre-se de que `voter_count` armazenamos o número de eleitores na eleição e que, para cada eleitor em nossa eleição, queremos contar um voto.
- Lembre-se de que, para um eleitor `i`, seu candidato de primeira escolha é representado por `preferences[i][0]`, seu candidato de segunda escolha por `preferences[i][1]`, etc.
- Lembre-se de que `candidate` `struct` existe um campo chamado `eliminated`, que será nulo `true` se o candidato tiver sido eliminado da eleição.
- Lembre-se de que `candidate` `struct` existe um campo chamado `votes`, que você provavelmente desejará atualizar para o candidato preferido de cada eleitor.
- Lembre-se de que, depois de votar no primeiro candidato não eliminado de um eleitor, você deve parar por aí, e não continuar votando nos outros candidatos da lista. Você pode interromper um loop antecipadamente usando `break` uma condição.

▼ **Complete a `print_winner`função**

Conclua a `print_winner` função.

- Se algum candidato tiver mais da metade dos votos, seu nome deverá ser impresso e a função deverá retornar `true`.
- Se ninguém venceu a eleição ainda, a função deve retornar `false`.

Ao escrever seu código, considere esta dica:

- Lembre-se de que `voter_count` armazena o número de eleitores na eleição. Dado isso, como você expressaria o número de votos necessários para vencer a eleição?

▼ **Complete a `find_min`função**

Conclua a `find_min` função.

- A função deve retornar o total mínimo de votos para qualquer candidato que ainda esteja na disputa eleitoral.

Ao escrever seu código, considere esta dica:

- Provavelmente, você vai querer percorrer a lista de candidatos para encontrar aquele que ainda está na disputa e que tem o menor número de votos. Que informações você deve acompanhar ao percorrer a lista de candidatos?

▼ **Complete a `is_tie`função**

Conclua a `is_tie` função.

- A função recebe um argumento `min`, que será o número mínimo de votos que qualquer candidato na eleição possui atualmente.
- A função deve retornar verdadeiro `true` se todos os candidatos restantes na eleição tiverem o mesmo número de votos e retornar falso `false` caso contrário.

Ao escrever seu código, considere esta dica:

- Lembre-se de que um empate ocorre se todos os candidatos restantes na eleição tiverem o mesmo número de votos. Observe também que a `is_tie` função recebe um argumento `min`, que é o menor número de votos que qualquer candidato possui atualmente. Como você poderia usar essa função `min` para determinar se a eleição terminou em empate (ou, inversamente, se não houve empate)?

▼ **Complete a `eliminate`função**

Conclua a `eliminate` função.

- A função recebe um argumento `min`, que será o número mínimo de votos que qualquer candidato na eleição possui atualmente.
- A função deve eliminar o(s) candidato(s) que possui (em) `min` um número suficiente de votos.

Passo a passo

Como testar

Certifique-se de testar seu código para garantir que ele funcione corretamente...

- Uma eleição com qualquer número de candidatos (até o limite `MAX` de `9`)
- Votar em um candidato pelo nome.
- Votos inválidos para candidatos que não constam na cédula eleitoral.
- Imprimir o vencedor da eleição, caso haja apenas um.
- Não eliminar ninguém em caso de empate entre todos os candidatos restantes.

Correção

```
check50 cs50/problems/2025/x/runoff
```

Estilo

```
style50 runoff.c
```

Como enviar

```
submit50 cs50/problems/2025/x/runoff
```