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)

Maré

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. 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. Charlie	2. Charlie	2. Alice	2. Alice	2. Alice
3. Bob	3. Bob	3. Charlie	3. Charlie	3. Charlie

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

Quem deveria ganhar esta eleição? Em uma 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. (Note que, se você estiver familiarizado com o sistema de votação por eliminação instantânea, Charlie também venceria nesse sistema). Alice, no entanto, poderia argumentar, com razão, que deveria ser a vencedora em vez de Charlie: afinal, dos nove eleitores, a maioria (cinco deles) preferiu Alice a Charlie, então a maioria das pessoas ficaria mais satisfeita com Alice como vencedora.

Nesta eleição, Alice é a chamada "vencedora de Condorcet": a pessoa que teria vencido qualquer confronto direto contra outro candidato. Se a eleição fosse apenas entre Alice e Bob, ou apenas entre Alice e Charlie, Alice teria vencido.

O método de votação Tideman (também conhecido como "pares classificados") é um método de votação por ordem de preferência que garante a obtenção do vencedor de Condorcet da eleição, caso exista um. Em um arquivo chamado `tideman.js`, localizado `tideman.c` em uma pasta chamada `tideman.js` `tideman`, crie um programa para simular uma eleição pelo método de votação Tideman.

Demonstração

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

Alice
$ ./tideman Alice Bob Charlie
Number of voters: 3
Rank 1: David
Invalid vote.
$ ./tideman
Usage: tideman [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/tideman.zip
```

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

Em seguida, execute

```
unzip tideman.zip
```

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

```
rm tideman.zip
```

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

Agora digite

```
cd tideman
```

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

```
tideman/ $
```

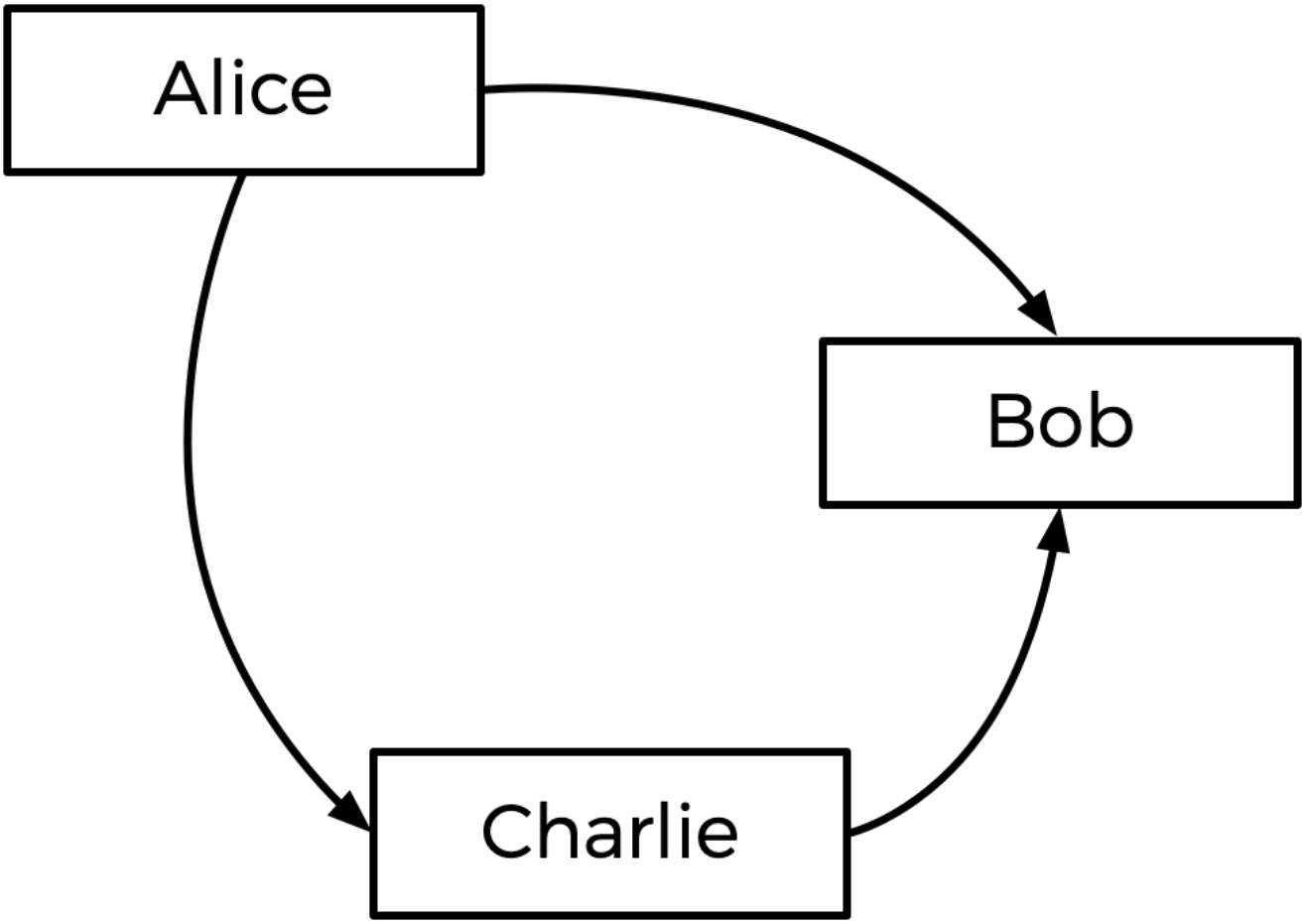
Se tudo correr bem, você deverá executar

```
ls
```

e veja um arquivo chamado `tideman.c`. A execução `code tideman.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!

Fundo

De forma geral, o método Tideman funciona construindo um "gráfico" de candidatos, onde uma seta (ou seja, uma aresta) do candidato A para o candidato B indica que o candidato A vence o candidato B em um confronto direto. O gráfico para a eleição acima, portanto, seria semelhante ao abaixo.



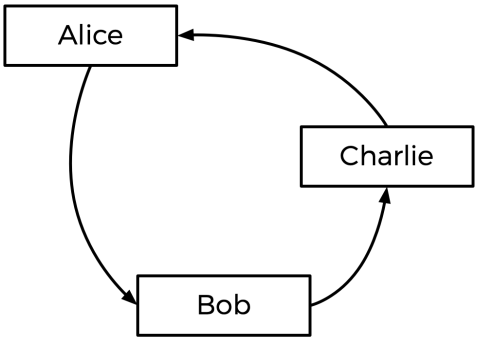
A seta de Alice para Bob significa que mais eleitores preferem Alice a Bob (5 preferem Alice, 4 preferem Bob). Da mesma forma, as outras setas significam que mais eleitores preferem Alice a Charlie, e mais eleitores preferem Charlie a Bob.

Analisando este gráfico, o método de Tideman afirma que o vencedor da eleição deve ser a "fonte" do gráfico (ou seja, o candidato que não possui nenhuma seta apontando para ele). Neste caso, a fonte é Alice — Alice é a única que não possui nenhuma seta apontando para ela, o que significa que ninguém é preferido a Alice em um confronto direto. Alice é, portanto, declarada a vencedora da eleição.

É possível, no entanto, que quando as setas forem desenhadas, não haja um vencedor de Condorcet. Considere as cédulas abaixo.

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

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



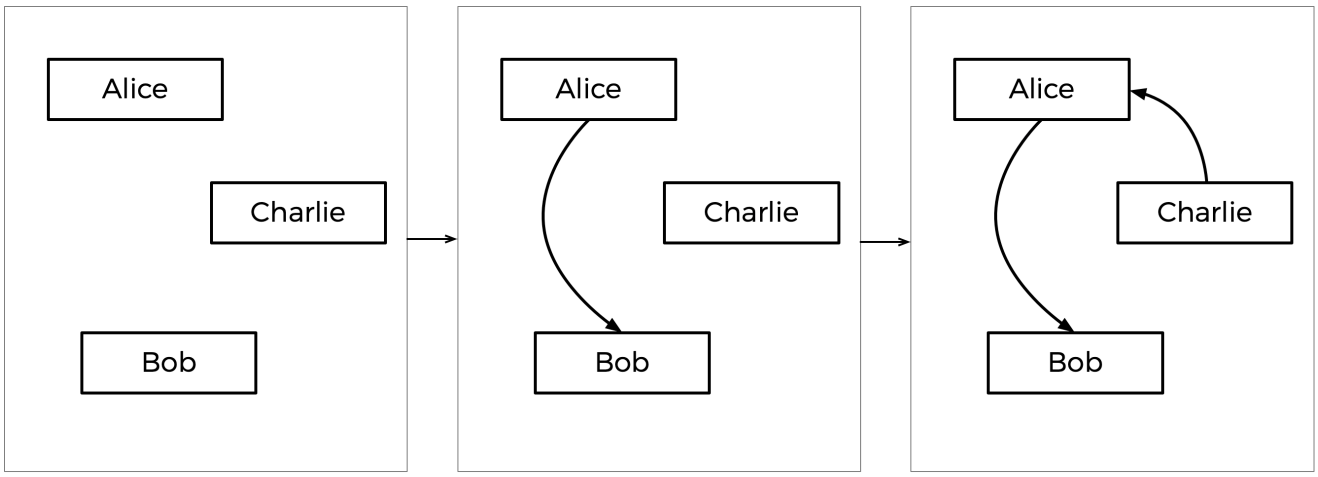
Entre Alice e Bob, Alice é preferida por uma margem de 7 a 2. Entre Bob e Charlie, Bob é preferido por uma margem de 5 a 4. Mas entre Charlie e Alice, Charlie é preferido por uma margem de 6 a 3. Se desenharmos o gráfico, não há uma fonte! Temos um ciclo de candidatos, onde Alice vence Bob, que vence Charlie, que vence Alice (muito parecido com um jogo de pedra-papel-tesoura). Nesse caso, parece não haver como escolher um vencedor.

Para lidar com isso, o algoritmo Tideman deve ter cuidado para evitar a criação de ciclos no grafo de candidatos. Como ele faz isso? O algoritmo fixa primeiro as arestas mais fortes, já que estas são indiscutivelmente as mais significativas. Em particular, o algoritmo Tideman especifica que as arestas de confronto devem ser "fixadas" no grafo uma de cada vez, com base na "força" da vitória (quanto mais pessoas preferirem um candidato ao seu oponente, mais forte será a vitória). Contanto que a aresta possa ser fixada no grafo sem criar um ciclo, ela é adicionada; caso contrário, a aresta é ignorada.

Como isso funcionaria no caso dos votos acima? Bem, a maior margem de vitória para um par é Alice vencendo Bob, já que 7 eleitores preferem Alice a Bob (nenhum outro confronto direto tem um vencedor preferido por mais de 7 eleitores). Portanto, a seta Alice-Bob é a primeira a ser inserida no gráfico. A segunda maior margem de vitória é a de Charlie, por 6 a 3, sobre Alice, então essa seta é a próxima a ser inserida.

A seguir, temos a vitória de Bob sobre Charlie por 5 a 4. Mas observe: se adicionássemos uma seta de Bob para Charlie agora, criaríamos um ciclo! Como o grafo não permite ciclos, devemos ignorar essa aresta e não adicioná-la ao grafo. Se houvesse mais setas a serem consideradas, as analisaríamos em seguida, mas essa era a última seta, então o grafo está completo.

Este processo passo a passo é mostrado abaixo, com o gráfico final à direita.



Com base no gráfico resultante, Charlie é a fonte (não há seta apontando para Charlie), portanto, Charlie é declarado o vencedor desta eleição.

Em termos mais formais, o método de votação Tideman consiste em três partes:

- **Apuração** : Depois que todos os eleitores tiverem indicado suas preferências, determine, para cada par de candidatos, quem é o candidato preferido e por qual margem.

- **Ordenar** : Ordene os pares de candidatos em ordem decrescente de força da vitória, onde a força da vitória é definida como o número de eleitores que preferem o candidato escolhido.
- **Bloqueio** : Começando pelo par mais forte, percorra os pares de candidatos em ordem e "bloqueie" cada par no grafo de candidatos, desde que o bloqueio desse par não crie um ciclo no grafo.

Uma vez que o grafo esteja completo, a origem do grafo (aquela que não possui arestas apontando para ela) é a vencedora!

Entendimento

Vamos dar uma olhada em `tideman.c`...

Primeiramente, observe a matriz bidimensional `preferences`. O número inteiro `preferences[i][j]` representará a quantidade de eleitores que preferem o candidato `i` em relação ao candidato `j`.

O arquivo também define outra matriz bidimensional, chamada `locked`, que representará o grafo candidato. `locked` é uma matriz booleana, portanto, `locked[i][j]` sendo `true` representa a existência de uma aresta apontando do candidato `i` para o candidato `j`; `false` significa que não há aresta. (Caso tenha curiosidade, essa representação de um grafo é conhecida como "matriz de adjacência").

A seguir, temos um `struct` chamado `pair`, usado para representar um par de candidatos: cada par inclui o `winner` índice do candidato de e o `loser` índice do candidato de .

Os próprios candidatos são armazenados na matriz `candidates`, que é uma matriz de `string`s representando os nomes de cada um dos candidatos. Há também uma matriz de `pairs`, que representará todos os pares de candidatos (para os quais um é preferido em relação ao outro) na eleição.

O programa também possui duas variáveis globais: `pair_count` e `candidate_count`, que representam o número de pares e o número de candidatos nas matrizes `pairs` e `candidates`, respectivamente.

Agora `main`, observe que, após determinar o número de candidatos, o programa percorre o `locked` grafo e inicialmente define todos os valores como 0 `false`, o que significa que nosso grafo inicial não terá arestas.

Em seguida, o programa percorre todos os eleitores e coleta suas preferências em um array chamado `ranks` (através de uma chamada a `vote`), onde `ranks[i]` é o índice do candidato que é a `i`-ésima preferência do eleitor. Essas classificações são passadas para a `record_preference` função , cuja função é pegar essas classificações e atualizar a `preferences` variável global.

Assim que todos os votos forem computados, os pares de candidatos são adicionados ao `pairs` array por meio de uma chamada a `add` `add_pairs`, ordenados por meio de uma chamada a `sort` `sort_pairs` e fixados no grafo por meio de uma chamada a `lock` `lock_pairs`. Finalmente, `print_winner` `print` é chamado para imprimir o nome do vencedor da eleição!

Mais adiante no arquivo, você verá que as funções `vote`, `record_preference`, `add_pairs`, `sort_pairs`, `lock_pairs`, e `print_winner` estão em branco. Isso fica a seu critério!

Especificação

Complete a implementação `tideman.c` de forma que simule uma eleição de Tideman.

- Conclua a `vote` função.
  - A função recebe os argumentos `rank`, `name`, e `ranks`. Se `name` for uma correspondência com o nome de um candidato válido, você deve atualizar a `ranks` matriz para indicar que o eleitor tem o candidato como sua `rank` preferência (onde `0` é a primeira preferência, `1` é a segunda preferência, etc.).
  - Lembre-se que aqui representa a preferência `ranks[i]` do usuário . `i`
  - A função deve retornar `true` se a classificação foi registrada com sucesso e, `false` caso contrário (se, por exemplo, `name` não for o nome de um dos candidatos).
  - Pode-se presumir que não haverá dois candidatos com o mesmo nome.
- Conclua a `record_preferences` função.
  - A função é chamada uma vez para cada eleitor e recebe como argumento o `ranks` array (lembrando que `ranks[i]` é a -ésima preferência do eleitor `i`, onde `ranks[0]` é a primeira preferência).
  - A função deve atualizar o `preferences` array global para adicionar as preferências do eleitor atual. Lembre-se de que isso `preferences[i][j]` deve representar o número de eleitores que preferem o candidato `i` em relação ao candidato `j`.
  - Pode-se presumir que cada eleitor classificará cada um dos candidatos.
- Conclua a `add_pairs` função.
  - A função deve adicionar ao `pairs` array todos os pares de candidatos em que um candidato é preferido. Um par de candidatos empatados (em que um não é preferido em relação ao outro) não deve ser adicionado ao array.
  - A função deve atualizar a variável global `pair_count` para o número de pares de candidatos. (Portanto, todos os pares devem ser armazenados entre `pairs[0]` e `pairs[pair_count - 1]`, inclusive).
- Conclua a `sort_pairs` função.
  - A função deve ordenar o `pairs` array em ordem decrescente de força da vitória, onde a força da vitória é definida como o número de eleitores que preferem o

candidato escolhido. Se vários pares tiverem a mesma força de vitória, você pode assumir que a ordem não importa.

- Conclua a `lock_pairs` função.
  - A função deve criar o `locked` grafo, adicionando todas as arestas em ordem decrescente de força de vitória, desde que a aresta não crie um ciclo.
- Conclua a `print_winner` função.
  - A função deve imprimir o nome do candidato que é a fonte do grafo. Pode-se assumir que não haverá mais de uma fonte.

Você não deve modificar nada `tideman.c` além das implementações das funções `vote` `__get__` `record_preferences`, `add_pairs` `__set__`, `sort_pairs` `__set__`, `lock_pairs` `__set__` e `__set__` `print_winner` (e a inclusão de arquivos de cabeçalho adicionais, se desejar). É permitido adicionar funções adicionais a `__set__` `tideman.c`, desde que você não altere as declarações de nenhuma das funções existentes.

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.
- Imprimindo o vencedor da eleição

Correção

```
check50 cs50/problems/2025/x/tideman
```

Estilo

```
style50 tideman.c
```

Como enviar

```
submit50 cs50/problems/2025/x/tideman
```