












Isso é CS50

Introdução do CS50 à Ciência da Computação

OpenCourseWare

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

David J. Malan (<https://cs.harvard.edu/malan/>)
malan@harvard.edu

 (<https://www.clubhouse.com/@davidjmalan>)  (<https://www.facebook.com/dmalan>) 
(<https://github.com/dmalan>)  (<https://www.instagram.com/davidjmalan/>) 
(<https://www.linkedin.com/in/malan/>)  (<https://orcid.org/0000-0001-5338-2522>) 
(<https://www.quora.com/profile/David-J-Malan>)  (<https://www.reddit.com/user/davidjmalan>) 
(<https://www.tiktok.com/@davidjmalan>)  (<https://davidjmalan.t.me/>) 
(<https://twitter.com/davidjmalan>)

Tideman

Para este programa, você implementará um programa que executa uma eleição do Tideman, conforme abaixo.

```
$ ./tideman Alice Bob Charlie
Number of voters: 5
Rank 1: Alice
Rank 2: Charlie
Rank 3: Bob

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

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

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

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

Charlie
```

Fundo

Você já conhece as eleições de pluralidade, que seguem um algoritmo muito simples para determinar o vencedor de uma eleição: cada eleitor recebe um voto, e o candidato com mais votos vence.

Mas o voto de pluralidade tem algumas desvantagens. O que acontece, por exemplo, em uma eleição com três candidatos, e as cédulas abaixo são lançadas?

Ballot	Ballot	Ballot	Ballot	Ballot
Alice	Alice	Bob	Bob	Charlie

Uma votação de pluralidade aqui declararia um empate entre Alice e Bob, já que cada um tem dois votos. Mas esse é o resultado certo?

Há outro tipo de sistema de votação, conhecido como sistema de votação por classificação. Em um sistema de escolha ranqueada, os eleitores podem votar em mais de um candidato. Em vez de apenas votar na primeira escolha, eles podem classificar os candidatos em ordem de preferência. As cédulas resultantes podem, portanto, parecer como abaixo.

Ballot	Ballot	Ballot	Ballot	Ballot
1. Alice 2. Bob 3. Charlie	1. Alice 2. Charlie 3. Bob	1. Bob 2. Alice 3. Charlie	1. Bob 2. Alice 3. Charlie	1. Charlie 2. Alice 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 agora pode ter um vencedor. A corrida foi originalmente empatada entre Alice e Bob. Mas o eleitor que escolheu Charlie preferiu Alice a Bob, então Alice poderia ser declarada a vencedora.

A votação por escolha classificada também pode resolver outra desvantagem potencial da votação por pluralidade. Confira as votações a seguir.

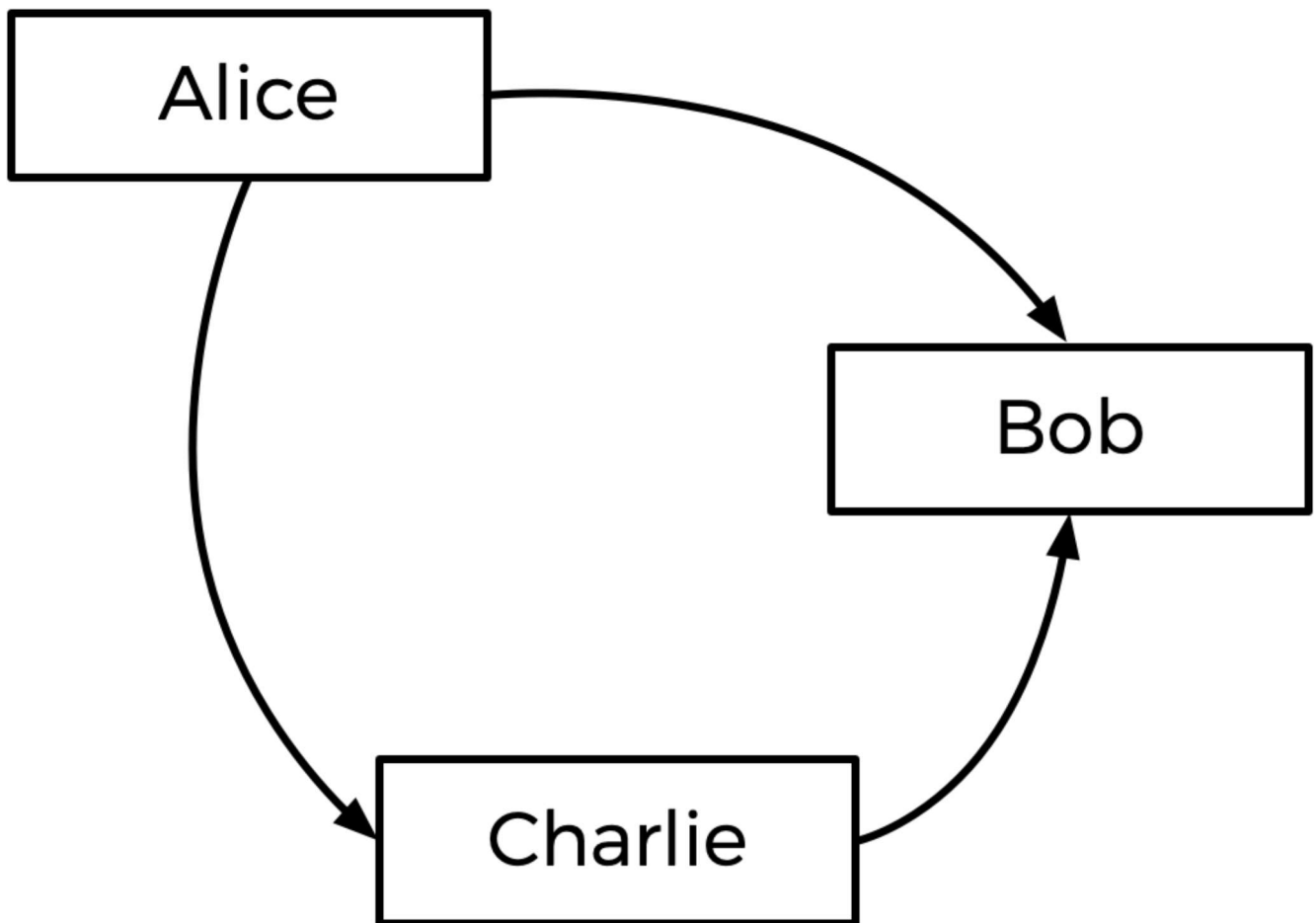
Ballot 1. Alice 2. Charlie 3. Bob	Ballot 1. Alice 2. Charlie 3. Bob	Ballot 1. Bob 2. Alice 3. Charlie	Ballot 1. Bob 2. Alice 3. Charlie	Ballot 1. Bob 2. Alice 3. Charlie
Ballot 1. Charlie 2. Alice 3. Bob	Ballot 1. Charlie 2. Alice 3. Bob	Ballot 1. Charlie 2. Alice 3. Bob	Ballot 1. Charlie 2. Bob 3. Alice	

Quem deve ganhar esta eleição? Em uma votação de pluralidade em que cada eleitor escolhe apenas sua primeira preferência, Charlie vence esta eleição com quatro votos em comparação com apenas três para Bob e dois para Alice. (Observe que, se você estiver familiarizado com o sistema de votação de segundo turno instantâneo, Charlie também vence aqui nesse sistema). Alice, no entanto, poderia razoavelmente argumentar que ela deveria ser a vencedora da eleição em vez de Charlie: afinal, dos nove eleitores, a maioria (cinco deles) preferia Alice a Charlie, então a maioria das pessoas ficaria mais feliz com Alice. como o vencedor em vez de Charlie.

Alice é, nesta eleição, a chamada “vencedora de Condorcet” da eleição: aquela que teria vencido qualquer confronto direto contra outro candidato. Se a eleição tivesse sido apenas Alice e Bob, ou apenas Alice e Charlie, Alice teria vencido.

O método de votação Tideman (também conhecido como “pares classificados”) é um método de votação de escolha classificada que garante a produção do Condorcet vencedor da eleição, se houver.

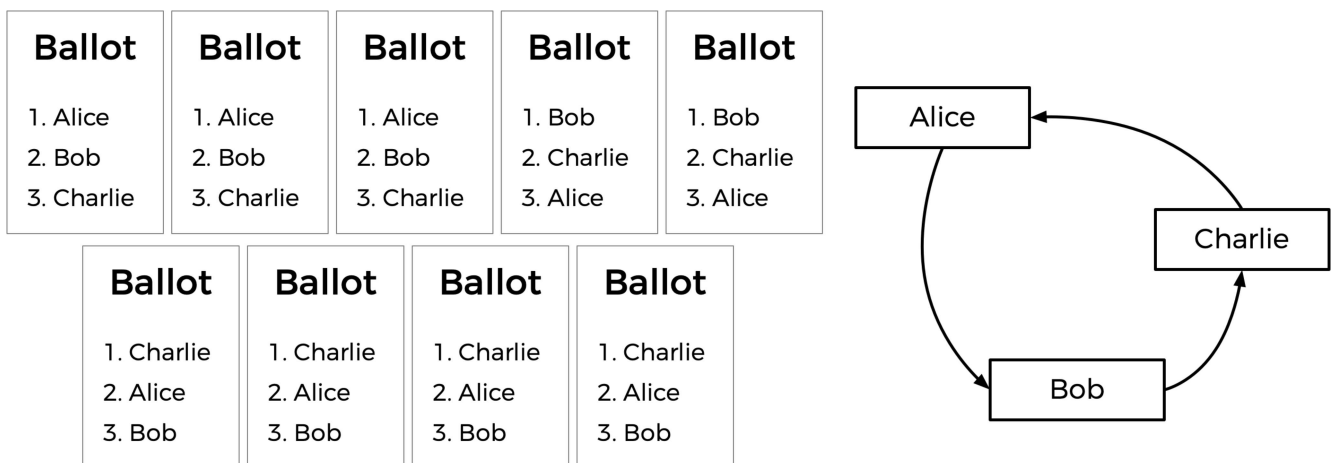
De um modo geral, o método Tideman funciona construindo um “gráfico” de candidatos, onde uma seta (ou seja, borda) do candidato A para o candidato B indica que o candidato A vence o candidato B em um confronto direto. O gráfico para a eleição acima, então, ficaria como o abaixo.



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.

Olhando para este gráfico, o método Tideman diz que o vencedor da eleição deve ser a “fonte” do gráfico (ou seja, o candidato que não tem seta apontando para ele). Nesse caso, a fonte é Alice — Alice é a única que não tem nenhuma seta apontando para ela, o que significa que ninguém tem preferência frente a frente com Alice. Alice é assim declarada a vencedora da eleição.

É possível, porém, que quando as flechas forem sorteadas, não haja um vencedor Condorcet. Considere as cédulas abaixo.



Entre Alice e Bob, Alice é preferida a Bob por uma margem de 7-2. Entre Bob e Charlie, Bob é preferido a Charlie por uma margem de 5-4. Mas entre Charlie e Alice, Charlie é preferido sobre Alice por uma margem de 6-3. Se desenharmos o gráfico, não há fonte! Temos um ciclo de candidatos, onde Alice vence Bob que

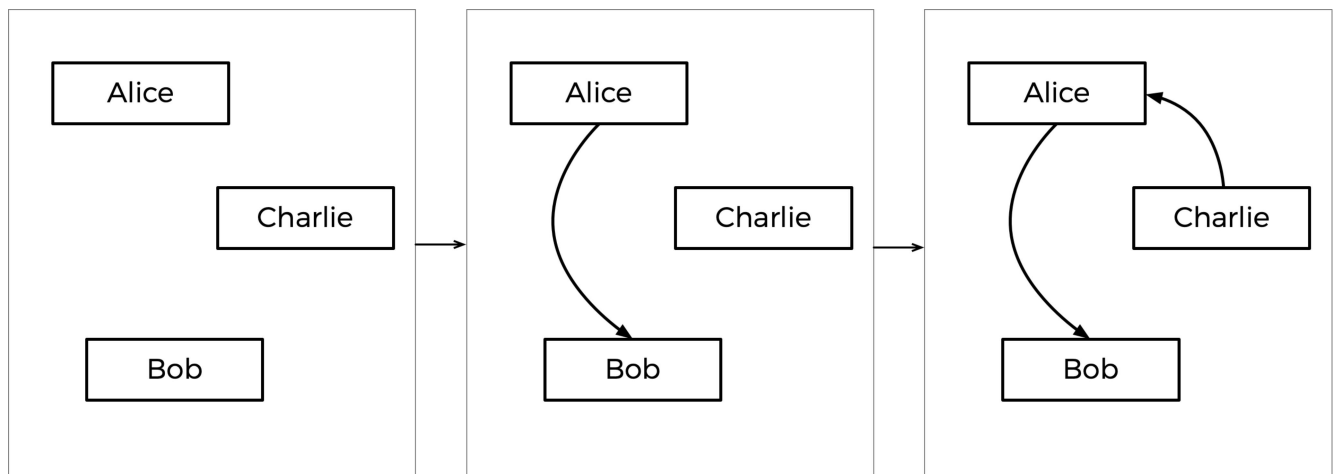
vence Charlie que vence Alice (muito parecido com um jogo de pedra-papel-tesoura). Nesse caso, parece que não há como escolher um vencedor.

Para lidar com isso, o algoritmo Tideman deve ter cuidado para evitar a criação de ciclos no grafo candidato. Como ele faz isso? O algoritmo bloqueia as arestas mais fortes primeiro, já que essas são indiscutivelmente as mais significativas. Em particular, o algoritmo Tideman especifica que as arestas de matchup devem ser “travadas” no gráfico uma de cada vez, com base na “força” da vitória (quanto mais pessoas preferirem um candidato ao invés de seu oponente, mais forte será a vitória). Desde que a aresta possa ser bloqueada no grafo sem criar um ciclo, a aresta 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 votantes preferem Alice a Bob (nenhum outro confronto direto tem um vencedor preferido por mais de 7 votantes). Portanto, a seta de Alice-Bob é bloqueada primeiro no gráfico. A próxima maior margem de vitória é a vitória de Charlie por 6-3 sobre Alice, de modo que a flecha é a próxima.

A seguir, a vitória de Bob por 5 a 4 sobre Charlie. Mas observe: se fôssemos adicionar uma flecha de Bob para Charlie agora, criaríamos um ciclo! Como o gráfico não pode permitir ciclos, devemos pular essa aresta e não adicioná-la ao gráfico. Se houvesse mais setas a serem consideradas, olharíamos para as próximas, mas essa era a última seta, então o gráfico 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), então Charlie é declarado o vencedor desta eleição.

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

- **Contagem** : Uma vez que todos os eleitores tenham indicado todas as suas preferências, determine, para cada par de candidatos, quem é o candidato preferido e por qual margem eles são os preferidos.
- **Sort** : Classifica os pares de candidatos em ordem decrescente de força de vitória, onde força de vitória é definida como o número de eleitores que preferem o candidato preferido.
- **Lock** : Começando com o par mais forte, percorra os pares de candidatos em ordem e “trave” cada par no grafo candidato, desde que o travamento desse par não crie um ciclo no grafo.

Assim que o gráfico estiver completo, a fonte do gráfico (aquele sem arestas apontando para ele) é o vencedor!

Começando

Faça login em [code.cs50.io \(https://code.cs50.io/\)](https://code.cs50.io), clique na janela do seu terminal e execute `cd` -o sozinho. Você deve descobrir que o prompt da janela do terminal é semelhante ao abaixo:

```
$
```

Próxima execução

```
wget https://cdn.cs50.net/2022/fall/psets/3/tideman.zip
```

para baixar um ZIP chamado `tideman.zip` em seu codespace.

Então execute

```
unzip tideman.zip
```

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

```
rm tideman.zip
```

e responda com “y” seguido de Enter no prompt para remover o arquivo ZIP que você baixou.

Agora digite

```
cd tideman
```

seguido de Enter para entrar (ou seja, abrir) nesse diretório. Seu prompt agora deve se parecer com o abaixo.

```
tideman/ $
```

Se tudo foi bem sucedido, você deve executar

```
ls
```

and see a file named `tideman.c`. Executing `code tideman.c` should open the file where you will type your code for this problem set. If not, retrace your steps and see if you can determine where you went wrong!

Understanding

Let's take a look at `tideman.c`.

First, notice the two-dimensional array `preferences`. The integer `preferences[i][j]` will represent the number of voters who prefer candidate `i` over candidate `j`.

The file also defines another two-dimensional array, called `locked`, which will represent the candidate graph. `locked` is a boolean array, so `locked[i][j]` being `true` represents the existence of an edge pointing from candidate `i` to candidate `j`; `false` means there is no edge. (If curious, this representation of a graph is known as an “adjacency matrix”).

Next up is a `struct` called `pair`, used to represent a pair of candidates: each pair includes the `winner`'s candidate index and the `loser`'s candidate index.

The candidates themselves are stored in the array `candidates`, which is an array of `strings` representing the names of each of the candidates. There's also an array of `pairs`, which will represent all of the pairs of candidates (for which one is preferred over the other) in the election.

The program also has two global variables: `pair_count` and `candidate_count`, representing the number of pairs and number of candidates in the arrays `pairs` and `candidates`, respectively.

Now onto `main`. Notice that after determining the number of candidates, the program loops through the `locked` graph and initially sets all of the values to `false`, which means our initial graph will have no edges in it.

Next, the program loops over all of the voters and collects their preferences in an array called `rank`s (via a call to `vote`), where `rank[i]` is the index of the candidate who is the `i`th preference for the voter. These ranks are passed into the `record_preference` function, whose job it is to take those ranks and update the global `preferences` variable.

Once all of the votes are in, the pairs of candidates are added to the `pairs` array via a call to `add_pairs`, sorted via a call to `sort_pairs`, and locked into the graph via a call to `lock_pairs`. Finally, `print_winner` is called to print out the name of the election's winner!

Further down in the file, you'll see that the functions `vote`, `record_preference`, `add_pairs`, `sort_pairs`, `lock_pairs`, and `print_winner` are left blank. That's up to you!

Specification

Complete the implementation of `tideman.c` in such a way that it simulates a Tideman election.

- Complete the `vote` function.
 - The function takes arguments `rank`, `name`, and `rank`s. If `name` is a match for the name of a valid candidate, then you should update the `rank`s array to indicate that the voter has the candidate as their `rank` preference (where `0` is the first preference, `1` is the second preference, etc.)
 - Recall that `rank[i]` here represents the user's `i`th preference.
 - The function should return `true` if the rank was successfully recorded, and `false` otherwise (if, for instance, `name` is not the name of one of the candidates).
 - You may assume that no two candidates will have the same name.
- Complete the `record_preferences` function.
 - The function is called once for each voter, and takes as argument the `rank`s array, (recall that `rank[i]` is the voter's `i`th preference, where `rank[0]` is the first preference).
 - The function should update the global `preferences` array to add the current voter's preferences. Recall that `preferences[i][j]` should represent the number of voters who prefer candidate `i` over candidate `j`.
 - You may assume that every voter will rank each of the candidates.
- Complete the `add_pairs` function.
 - The function should add all pairs of candidates where one candidate is preferred to the `pairs` array. A pair of candidates who are tied (one is not preferred over the other) should not be added to the array.

- The function should update the global variable `pair_count` to be the number of pairs of candidates. (The pairs should thus all be stored between `pairs[0]` and `pairs[pair_count - 1]`, inclusive).
- Complete the `sort_pairs` function.
 - The function should sort the `pairs` array in decreasing order of strength of victory, where strength of victory is defined to be the number of voters who prefer the preferred candidate. If multiple pairs have the same strength of victory, you may assume that the order does not matter.
- Complete the `lock_pairs` function.
 - The function should create the `locked` graph, adding all edges in decreasing order of victory strength so long as the edge would not create a cycle.
- Complete the `print_winner` function.
 - The function should print out the name of the candidate who is the source of the graph. You may assume there will not be more than one source.

You should not modify anything else in `tideman.c` other than the implementations of the `vote`, `record_preferences`, `add_pairs`, `sort_pairs`, `lock_pairs`, and `print_winner` functions (and the inclusion of additional header files, if you'd like). You are permitted to add additional functions to `tideman.c`, so long as you do not change the declarations of any of the existing functions.

Walkthrough



Usage

Your program should behave per the example below:

```
./tideman Alice Bob Charlie
Number of voters: 5
Rank 1: Alice
Rank 2: Charlie
```



```
Rank 3: Bob
```

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

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

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

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

```
Charlie
```

Testing

Be sure to test your code to make sure it handles...

- An election with any number of candidate (up to the `MAX` of `9`)
- Voting for a candidate by name
- Invalid votes for candidates who are not on the ballot
- Printing the winner of the election

Execute o abaixo para avaliar a exatidão do seu código usando `check50`. Mas certifique-se de compilar e testar você mesmo também!

```
check50 cs50/problems/2023/x/tideman
```

Execute o abaixo para avaliar o estilo do seu código usando `style50`.

```
style50 tideman.c
```

Como enviar

Em seu terminal, execute o abaixo para enviar seu trabalho.

```
submit50 cs50/problems/2023/x/tideman
```

