Data structures and algorithms

Tutorial 9

Amr Keleg

Faculty of Engineering, Ain Shams University

April 28, 2020

Contact: amr_mohamed@live.com

Outline

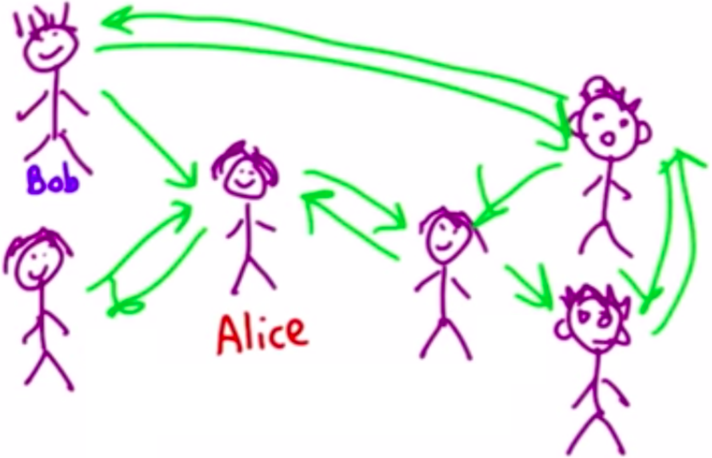
Outline

PageRank (PR) is:

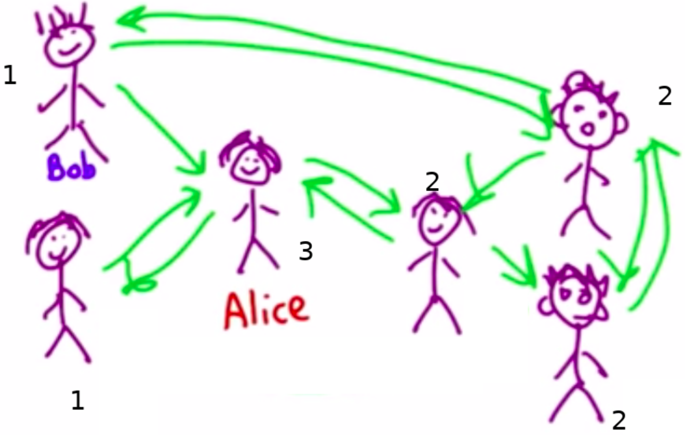
- an algorithm used by Google Search to rank web pages in their search engine results.
- named after Larry Page (one of the founders of Google).

- How to rank the web pages?
- How to give scores to the webpages?

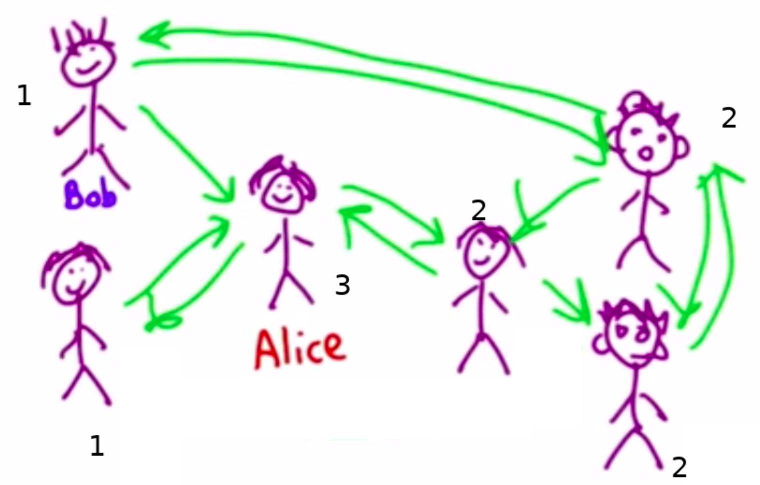
School/University Analogy:
Who is the most popular one?
Note: The fact that A is a friend of B doesn't imply that B is a friend of A.

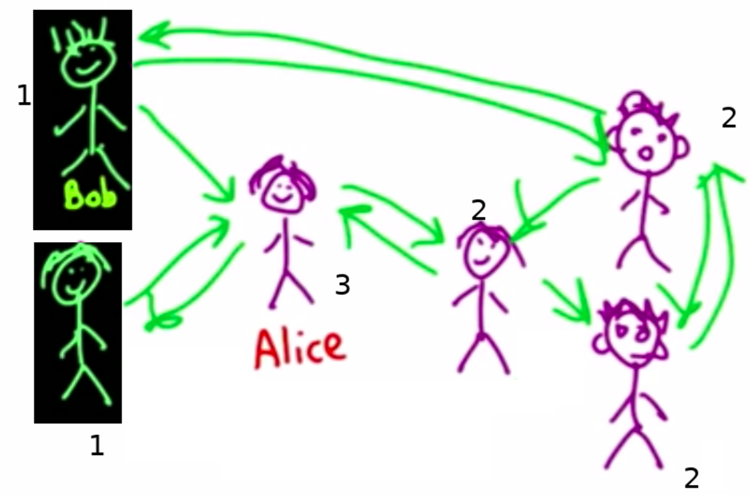


First definition of popularity:
The popularity of student A is proportional to the number of students who consider A to be their friend.
Popularity(A) = No of students who consider A to be their friend



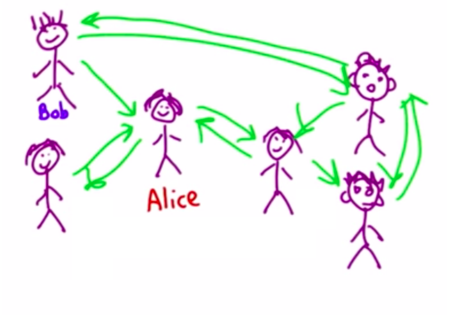
Can we do better?




1
Bob
1
3
Alice
2
2
2

- If you are a friend to non-popular people then you aren't popular.
- If you are a friend of popular people then you are popular.

- If you are a friend to non-popular people then you aren't popular.
- If you are a friend of popular people then you are popular.
- If you are the only friend to someone then you should get higher scores.

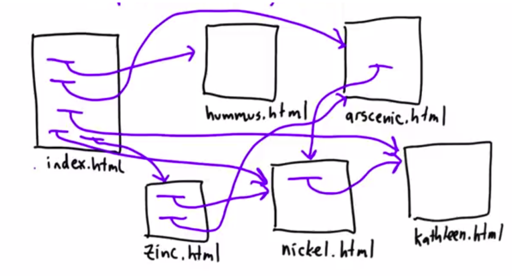


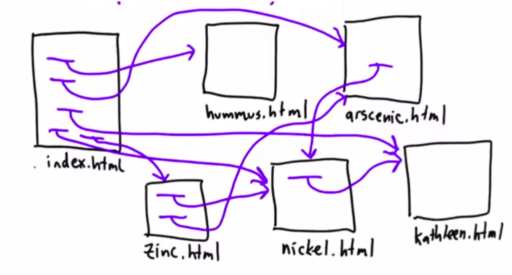
So, don't just count the number of incoming edges, Give weight to these edges.

$$Popularity(A) = \sum_{s \in B_A} \frac{Popularity(s)}{Ns}$$

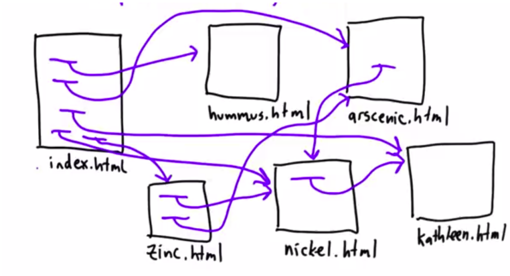
- B_A is the set of students who consider A to be their friend.
- Ns is the no of students that s consider to be their friend.

The same idea applies for web pages:
`https://udacity.github.io/cs101x/urank/`


hummus.html
arscenic.html
index.html
zinc.html
nickel.html
kathleen.html

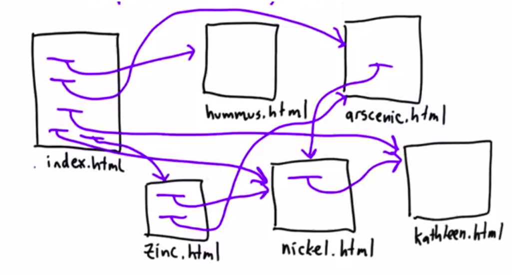


PageRank(hummus.html)=??
PageRank(aresenic.html)=??



PageRank(hummus.html)=??
PageRank(aresenic.html)=??

PageRank(hummus.html) = PageRank(index.html) / 5
PageRank(aresenic.html) = PageRank(index.html) / 5 + PageRank(zinc.html) /2



PageRank(hummus.html)=??
PageRank(aresenic.html)=??

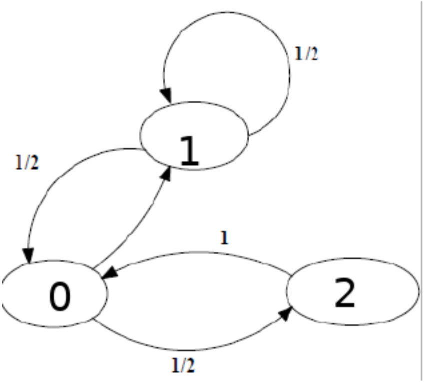
PageRank(hummus.html) = PageRank(index.html) / 5
PageRank(aresenic.html) = PageRank(index.html) / 5 + PageRank(zinc.html) /2

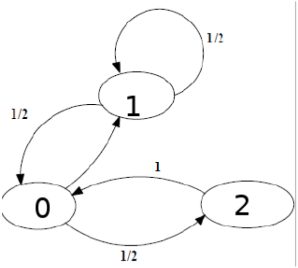
How to compute these dependent values?

Solution:

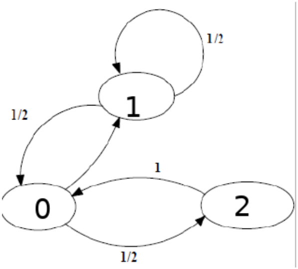
- Start with a guess for the PageRank for each page.
- Recompute the PageRank given the definition.
- Continue until PageRanks start to converge (don't change).

How to calculate the page rank for the following pages?



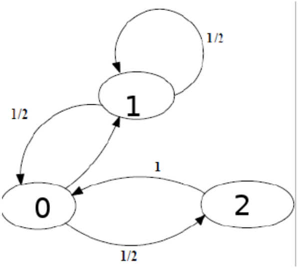


Initially, PageRank[0, it=0] = PageRank[1, it=0] = PageRank[2, it=0] = 1/3



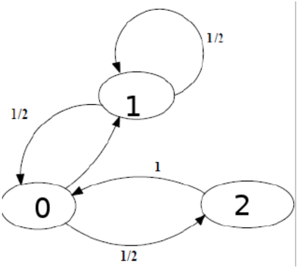
Initially, PageRank[0, it=0] = PageRank[1, it=0] = PageRank[2, it=0] = 1/3

- PageRank[0, it=1] =



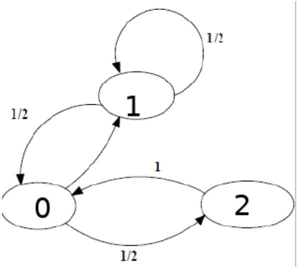
Initially, PageRank[0, it=0] = PageRank[1, it=0] = PageRank[2, it=0] = 1/3

- PageRank[0, it=1] = (PageRank[1, it=0]/2) + (PageRank[2, it=0]/1)



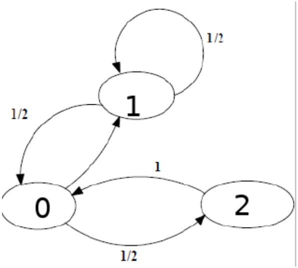
Initially, PageRank[0, it=0] = PageRank[1, it=0] = PageRank[2, it=0] = 1/3

- PageRank[0, it=1] = (PageRank[1, it=0]/2) + (PageRank[2, it=0]/1)
- PageRank[1, it=1] =



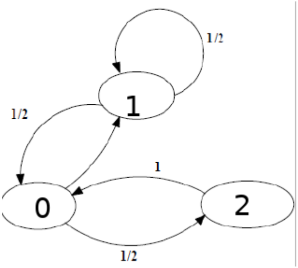
Initially, PageRank[0, it=0] = PageRank[1, it=0] = PageRank[2, it=0] = 1/3

- PageRank[0, it=1] = (PageRank[1, it=0]/2) + (PageRank[2, it=0]/1)
- PageRank[1, it=1] = (PageRank[0, it=0]/2) + (PageRank[1, it=0]/2)



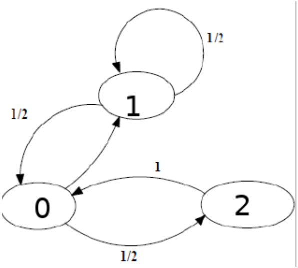
Initially, PageRank[0, it=0] = PageRank[1, it=0] = PageRank[2, it=0] = 1/3

- PageRank[0, it=1] = (PageRank[1, it=0]/2) + (PageRank[2, it=0]/1)
- PageRank[1, it=1] = (PageRank[0, it=0]/2) + (PageRank[1, it=0]/2)
- PageRank[2, it=1] =



Initially, PageRank[0, it=0] = PageRank[1, it=0] = PageRank[2, it=0] = 1/3

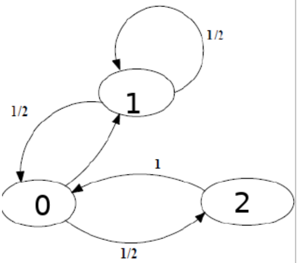
- PageRank[0, it=1] = (PageRank[1, it=0]/2) + (PageRank[2, it=0]/1)
- PageRank[1, it=1] = (PageRank[0, it=0]/2) + (PageRank[1, it=0]/2)
- PageRank[2, it=1] = (PageRank[0, it=0]/2)



Initially, PageRank[0, it=0] = PageRank[1, it=0] = PageRank[2, it=0] = 1/3

- PageRank[0, it=1] = (PageRank[1, it=0]/2) + (PageRank[2, it=0]/1)
- PageRank[1, it=1] = (PageRank[0, it=0]/2) + (PageRank[1, it=0]/2)
- PageRank[2, it=1] = (PageRank[0, it=0]/2)

$$\begin{bmatrix} PageRank[0, it = 1] \\ PageRank[1, it = 1] \\ PageRank[2, it = 1] \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} PageRank[0, it = 0] \\ PageRank[1, it = 0] \\ PageRank[2, it = 0] \end{bmatrix}$$



Initially, PageRank[0, it=0] = PageRank[1, it=0] = PageRank[2, it=0] = 1/3

- PageRank[0, it=1] = (PageRank[1, it=0]/2) + (PageRank[2, it=0]/1)
- PageRank[1, it=1] = (PageRank[0, it=0]/2) + (PageRank[1, it=0]/2)
- PageRank[2, it=1] = (PageRank[0, it=0]/2)

$$\begin{bmatrix} PageRank[0, it=1] \\ PageRank[1, it=1] \\ PageRank[2, it=1] \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} PageRank[0, it=0] \\ PageRank[1, it=0] \\ PageRank[2, it=0] \end{bmatrix}$$

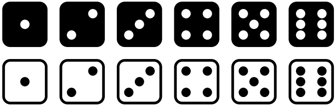
$$\begin{bmatrix} PageRank[0, it=1] \\ PageRank[1, it=1] \\ PageRank[2, it=1] \end{bmatrix} = \begin{bmatrix} 0 & 0.5 & 1 \\ 0.5 & 0.5 & 0 \\ 0.5 & 0 & 0 \end{bmatrix} \begin{bmatrix} PageRank[0, it=0] \\ PageRank[1, it=0] \\ PageRank[2, it=0] \end{bmatrix}$$

```python
import numpy as np

p = np.array([[1/3, 1/3, 1/3]]).T
M = np.array([[  0, 0.5, 1],
          [0.5, 0.5, 0],
          [0.5,   0, 0]])

for iteration in range(1, 16):
  p = np.matmul(M, p)
  print('After', iteration, 'iterations:', p.T)
```

After 0 iterations: [[0.33333333 0.33333333 0.33333333]]
After 1 iterations: [[0.5 0.33333333 0.16666667]]
After 2 iterations: [[0.33333333 0.41666667 0.25]]
After 3 iterations: [[0.45833333 0.375 0.16666667]]
After 4 iterations: [[0.35416667 0.41666667 0.22916667]]
After 5 iterations: [[0.4375 0.38541667 0.17708333]]
After 6 iterations: [[0.36979167 0.41145833 0.21875]]
After 7 iterations: [[0.42447917 0.390625 0.18489583]]
After 8 iterations: [[0.38020833 0.40755208 0.21223958]]
After 9 iterations: [[0.41601562 0.39388021 0.19010417]]
After 10 iterations: [[0.38704427 0.40494792 0.20800781]]
After 11 iterations: [[0.41048177 0.39599609 0.19352214]]
After 12 iterations: [[0.39152018 0.40323893 0.20524089]]
After 13 iterations: [[0.40686035 0.39737956 0.19576009]]
After 14 iterations: [[0.39444987 0.40211995 0.20343018]]
After 15 iterations: [[0.40449015 0.39828491 0.19722493]]

Outline

1 PageRank
- What is PageRank
- Probablilites fast RECAP
- Formal definition of PageRank - Random Surfer Model
- Formal definition of PageRank - Modified Surfer Model



For a fair dice, what is the probability that you get a 5?

How to calculate the probability of getting a 5?

- Get a dice.
- Roll it for a very very large number of attempts.
- Count the number of times you get a 5 and divide it by the number of trials.
- Voila!

Let's simulate it.

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int no_of_times[7] = {};
    int no_of_trials = 1000000;
    for (int i=0; i< no_of_trials; i++){
        int dice_no = 1 + (rand() % 6);
        no_of_times[dice_no]++;
    }

    for (int i=1; i<=6; i++){
        cout<<"P("<<i<<") is "<<
            1.0*no_of_times[i]/no_of_trials<<endl;
    }

    return 0;
}
```

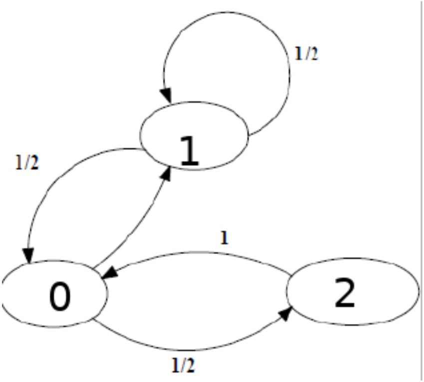
- $P(1)$ is 0.166511
- $P(2)$ is 0.166655
- $P(3)$ is 0.167279
- $P(4)$ is 0.166835
- $P(5)$ is 0.166365
- $P(6)$ is 0.166355

Outline

- A surfer moves through the Internet randomly.
- At first, They enter a URL.
- Then,they follow a series of successive links for a very long time.
- In a random surfer model, it is assumed that the link which is clicked next is selected at random.

The page rank of Page P is the probability that the random surfer will end the walk at page P.

How to calculate the page rank for the following pages?



Simulation code:

```
import random

nodes = [0, 1, 2]
edges = {0: [1, 2], 1:[0, 1], 2:[0]}
no_of_times = {}

for node in nodes:
  no_of_times[node] = 0

def random_walk(node, timestamp, limit):
  if timestamp == limit:
    no_of_times[node] += 1
  else:
    next_node_indx = random.randint(0, len(edges[node])-1)
    random_walk(edges[node][next_node_indx], timestamp +1, limit)

if __name__ == '__main__':
  no_of_walks = 100000
  max_walk_length = 10

  for _ in range(no_of_walks):
    start_node = random.randint(0, len(nodes)-1)
    random_walk(start_node, 0, max_walk_length)

  for node in nodes:
    no_of_times[node] /= no_of_walks

  for node in nodes:
    print('Probability of ending at node ({}) is: {}'.format(node, no_of_times[node]))
```

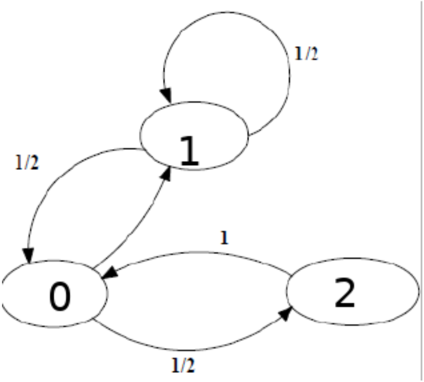
Simulation results:

- Probability of ending at node (0) is: 0.38572
- Probability of ending at node (1) is: 0.40654
- Probability of ending at node (2) is: 0.20774

Formal Definition:

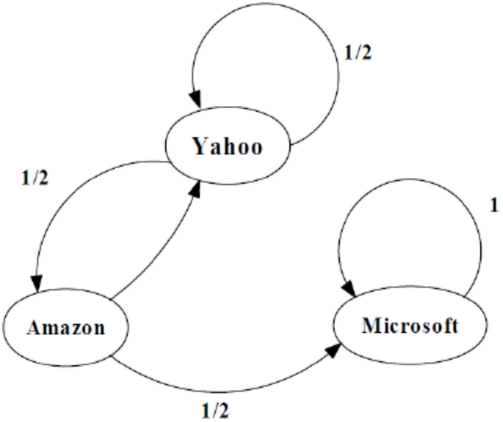
- P(start at page 0) $= 1/3$
- P(start at page 1) $= 1/3$
- P(start at page 2) $= 1/3$

What is the probability that the surfer is at page 1 after one click?



- P(page 0, time t) = 0.5 * P(page 1, time t-1) + 1 * P(page 2, time t-1)
- P(page 1, time t) = 0.5 * P(page 0, time t-1) + 0.5 * P(page 1, time t-1)
- P(page 2, time t) = 0.5 * P(page 0, time t-1)

Defects in the model:



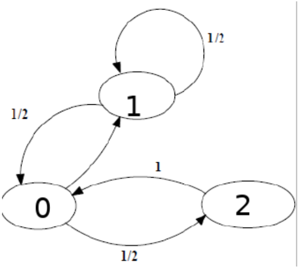
Outline

- A surfer moves through the Internet randomly.
- At first, They enter a URL.
- Then,they may follow a series of successive links or use a bookmark to go directly to a webpage.
- In a random surfer model, it is assumed that the link which is clicked next is selected at random.

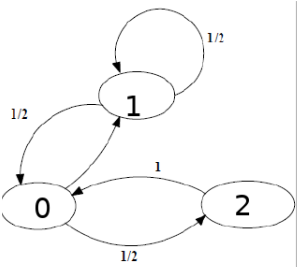
Probability that the user continues surfing is d.
Probability that the user uses a bookmark is 1-d.

Formal Definition:

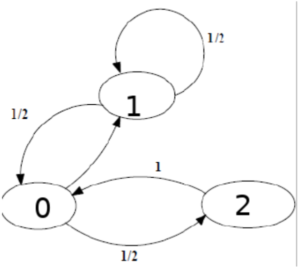
- P(start at page 0) $= 1/3$
- P(start at page 1) $= 1/3$
- P(start at page 2) $= 1/3$



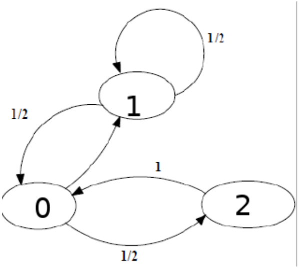
What is the probability that the surfer is at page 0 at time t?



What is the probability that the surfer is at page 0 at time t?
P(page 0, time t) = 0.5 * P(page 1, time t-1)



What is the probability that the surfer is at page 0 at time t?
P(page 0, time t) = 0.5 * P(page 1, time t-1)
How to model the direct navigation to a certain link?



What is the probability that the surfer is at page 0 at time t?

$P(\text{page } 0, \text{time } t) = 0.5 * P(\text{page } 1, \text{time } t\text{-}1)$

How to model the direct navigation to a certain link?

$P(\text{page } 0, \text{time } t) = d * (0.5 * P(\text{page } 1, \text{time } t\text{-}1) + 1 * P(\text{page } 2, \text{time } t\text{-}1)) + (1\text{-}d) * (1/N)$

The final page ranks are:

- $\text{PR(Amazon)} = 7/33$
- $\text{PR(Yahoo)} = 5/33$
- $\text{PR(Microsoft)} = 21/33$

Feedback form: `https://forms.gle/tK9hQEvKD1guD5vf6`